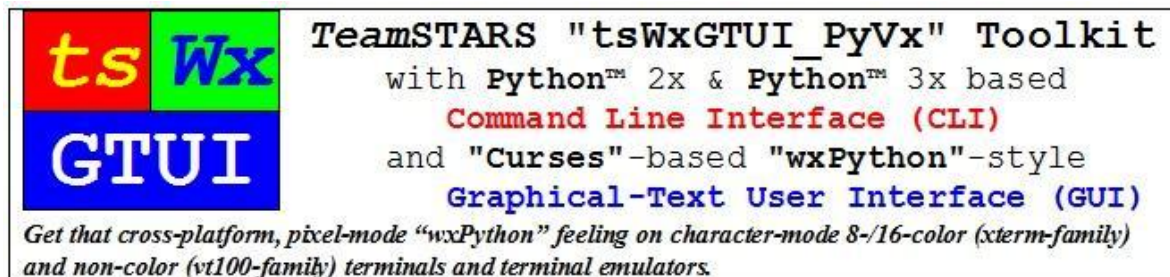


Software Development Plan

Vol. 4 - "tsWxGTUI_PyVx" Toolkit

Rev. 0.0.6 (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) 2010-2016 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_Repository/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)	10
2	DEFINITIONS, ABBREVIATIONS, AND ACRONYMS	23
3	OVERVIEW OF REQUIRED WORK	51
3.1	Purpose	51
3.2	Requirements and Constraints (Development Plan)	52
3.2.1	Goals & Capabilities (Development Plan)	53
3.2.2	Non-Goals & Limitations (Development Plan)	82
3.2.3	Comparison of wxPython with tsWxGTUI (Development Plan)	83
3.3	Concept	86
3.3.1	Technologies	92
3.3.2	Design	94
3.3.3	Architecture	111
4	PLANS FOR PERFORMING GENERAL SOFTWARE DEVELOPMENT ACTIVITIES	121
4.1	Software Development Process	121
4.1.1	Prototype	122
4.1.2	Pre-Alpha	123
4.1.3	Alpha	124
4.1.4	Beta	124
4.1.5	Release Candidate	125
4.1.6	Release	125
4.2	General Plans For Software Development	125
4.2.1	Software Development Methods	126
4.2.2	Standards For Software Products	127
4.2.3	Reusable Software Products	147
4.2.4	Handling Of Critical Requirements	149
4.2.5	Computer Hardware Resource Utilization	150
4.2.6	Recording Rationale	150
4.2.7	Access For Acquirer Review	150
5	PLANS FOR PERFORMING DETAILED SOFTWARE DEVELOPMENT ACTIVITIES	151
5.1	Project Planning And Oversight	152
5.1.1	Software Development Planning	153
5.1.2	CSCI Test Planning	153
5.1.3	System Test Planning	153
5.1.4	Software Installation Planning	153
5.1.5	Software Transition Planning	153

5.1.6	Following and Updating Plans.....	153
5.2	Establishing A Software Development Environment.....	153
5.3	System Requirements Analysis	154
5.4	Software Design	154
5.5	Software Implementation And Unit Testing	155
5.6	Unit Integration And Testing	155
5.7	CSCI Qualification Testing	156
5.8	CSCI/HWCI Integration And Testing	156
5.9	System Qualification Testing	157
5.10	Preparing For Software Use.....	158
5.11	Preparing For Software Transition.....	159
5.12	Software Configuration Management	160
5.13	Software Product Evaluation.....	160
5.14	Software Quality Assurance.....	161
5.15	Corrective Action.....	161
5.16	Joint Technical and Management Reviews	162
5.17	Other Software Development Activities	162

6 SCHEDULES AND ACTIVITY NETWORK 163

6.1	Schedule(s).....	163
6.2	Activity Network.....	163

7 PROJECT ORGANIZATION AND RESOURCES 165

7.1	Project Organization.....	165
7.2	Project Resources	166
7.2.1	Personnel Resources	167
7.2.2	Overview of Developer Facilities	167
7.2.3	Acquirer-furnished Equipment, Software, Services, Documentation, Data, and Facilities.....	167
7.2.4	Other Required Resources	168

8 NOTES 169

8.1	Toolkit Evolution Plans & Design Considerations.....	169
8.1.1	Terminal Control Library Considerations.....	170
8.1.2	Operator Terminal Design Considerations	172
8.1.3	Computer Processor Design Considerations.....	172
8.1.4	Python Interpreter and Virtual Machine Considerations	173
8.1.5	Python Application Program Launch Considerations.....	174
8.1.6	TeamSTARS "tsWxGTUI_PyVx" Toolkit Considerations	175
8.1.7	Reference Documentation.....	175

9 APPENDIXES 177

10 TO-DO-LIST 179

10.1	New Features:	179
10.2	Modifications:	180
10.3	Troubleshooting:	180
10.4	Validation/Regression Test:	181

11 SAMPLE SCREEN SHOTS 183

11.1	test_tsWxLinesOfCode Application using xterm via Terminal on Mac OS X	184
11.2	test_tsWxRSM Application using xterm via Terminal on Mac OS X	185
11.3	test_tsWxStaticBoxSizer Application using xterm via Terminal on Mac OS X	186
11.4	test_tsWxWidgets Application using vt100 via Terminal on Mac OS X	187
11.5	test_tsWxWidgets Application using xterm via iTerm on Mac OS X	188
11.6	test_tsWxWidgets Application using xterm via Terminal on Mac OS X	189
11.7	test_tsWxWidgets Application using xterm via Terminal on Mac OS X	190
11.8	test_tsWxMarkupDiagnostics via Cygwin-Mintty Terminal on Mac OS X	191

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 10)

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 the following Python language specific versions:</p> <ul style="list-style-type: none"> ▪ TeamSTARS "tsWxGTUI_Py1x" Toolkit (reserved for legacy Python 1x; to be created by back-port from TeamSTARS "tsWxGTUI_Py2x") ▪ TeamSTARS "tsWxGTUI_Py2x" Toolkit (for mature Python 2x; once created as the Toolkit prototype; then upgraded as Python 2x evolves) ▪ TeamSTARS "tsWxGTUI_Py3x" Toolkit (for evolving Python 3x; once created as port from TeamSTARS "tsWxGTUI_Py2x; then upgraded as Python 3x evolves) ▪ TeamSTARS "tsWxGTUI_Py4x" Toolkit (reserved for future Python 4x; to be created by port from TeamSTARS "tsWxGTUI_Py3x"; then upgraded as Python 4x evolves)
Title	TeamSTARS "tsWxGTUI_PyVx" Toolkit with Python 2x & Python 3x based Command Line Interface (CLI) and "Curses"-based "wxPython"-style, Graphical-Text User Interface (GUI)
Identification Number	N/A

Version Number	0.0.6
Release Number	0.0.6 (pre-alpha)
Build Date	01/10/2016

Draft

**Usage Terms and
Conditions --- Key
Points**

This is free and open source software. The *TeamSTARS* "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.

Please note the following:

- 1 Each *TeamSTARS* "tsWxGTUI_PyVx" Toolkit distribution includes a root directory whose name ("tsWxGTUI") has a suffix "_PyVx" that reflects the associated Python language syntax version:
 - a) **"tsWxGTUI_Py1x"** - *Reserved* 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 2.0.0-2.7.11**.
 - c) **"tsWxGTUI_Py3x"** - Root of third generation syntax files and subdirectories for **Python 3.0.0-3.5.1**.
 - d) **"tsWxGTUI_Py4x"** - *Reserved* for future Root of next generation syntax files and subdirectories for Python 4.x.
- 2 You can use, study, modify and redistribute the distribution only under the *Terms and Conditions* files provided in the *"/tsWxGTUI_PyVx/Documents"* sub-directory:
 - a) **"Copyright.txt"** - Attribution for the principal Author(s) of intellectual property created specifically for the *TeamSTARS* "tsWxGTUI_PyVx" Toolkit.
 - b) **"Credits.txt"** - Attribution for those third-party Author(s) whose intellectual property has been adapted for use in the *TeamSTARS* "tsWxGTUI_PyVx" Toolkit.
 - c) **"License.txt"** - Statements of rights, obligations and limitations stipulated by the Authors of intellectual property included in the *TeamSTARS* "tsWxGTUI_PyVx" Toolkit.
 - d) **"Notices.txt"** - Announcement calling the *TeamSTARS* "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's cross-platform design facilitates the creation, enhancement, troubleshooting, maintenance, porting and support of:

- 1 Mission-critical equipment used by commercial, industrial, medical and military customers.
Such equipment can include a broad range of embedded computer systems which satisfy the Platform Hardware and Software Requirements.
- 2 Application programs used for the local and remote supervisory control and data acquisition associated with automation, communication, control, diagnostic, instrumentation and simulation.
Such application software typically includes "operator-friendly" user interfaces.

The *TeamSTARS* "tsWxGTUI_PyVx" Toolkit includes 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 for the first generation Python language 1.0.0-1.6.1 (reserved for future back-port from *TeamSTARS* "tsWxGTUI_Py2x" Toolkit)
 - b) *TeamSTARS* "tsWxGTUI_Py2x" Toolkit for the second generation Python language 2.0.0-2.7.11
 - c) *TeamSTARS* "tsWxGTUI_Py3x" Toolkit for the third generation Python language 3.0.0-3.5.1
 - d) *TeamSTARS* "tsWxGTUI_Py4x" Toolkit for the fourth generation Python language 4.0.0 (reserved for future port from *TeamSTARS* "tsWxGTUI_Py3x" Toolkit)
 - e) *TeamSTARS* "tsWxGTUI_PyVx" Toolkit reserved for distribution 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. Mission-critical equipment typically includes embedded computer systems which 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 thereof 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	<p>This document alerts potential and existing users of the availability, capabilities, limitations and sources for the latest source code release and technical support.</p> <p>Included are the following topics:</p> <p>1 ANNOUNCEMENT</p> <p>a) About</p> <p>Platform Hardware and Software Requirements</p> <p>What is the toolkit designed to do?</p> <p>What is included in the release?</p> <p>How might you use the Toolkit?</p> <p>Where to get further infomation?</p>
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-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> <p>1 INTRODUCTION</p> <p>a) About</p> <p>Platform Hardware and Software Requirements</p> <p>What is the toolkit designed to do?</p> <p>What is included in the release?</p> <p>How might you use the Toolkit?</p> <p>Where to get further information?</p> <p>b) System Block Diagrams</p> <p>Block Diagram</p> <p>Stand Alone System Architecture</p> <p>Stand Among System Architecture</p> <p>c) Usage Terms & Conditions</p> <p>2 SCREENSHOTS</p> <p>a) Latest XTERM & VT100 Desktops</p> <p>b) Earlier XTERM Terminal Emulator with 8-Color / 64-Color Pairs</p> <p>c) Earlier VT100 Terminal Emulator with 1-Color / 2-Color Pairs</p>
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> <p>1 SCOPE (System Specification)</p> <p>a) Identification (System Specification)</p> <p>b) Purpose (System Specification)</p> <p>c) Document Overview (System Specification)</p> <p>2 SYSTEM OVERVIEW (tsWxGTUI Introduction)</p> <p>a) Purpose of System and Software</p> <p>b) General Nature of the System and Software</p>

		<ul style="list-style-type: none"> 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 Technology f) Appendix F - History of System Development. Operation, and Maintenance
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> <ul style="list-style-type: none"> a) Copyright - Identifies the original author(s) of source code and documentation for building block libraries and application programs. b) License - Identifies the original author's rules for using, modifying and redistributing source code and documentation for building block libraries and application programs. c) Notices - Identifier placed on copies of the source code and documentation to inform the world of copyright ownership and the applicable license. d) Splash Screen Designer's Guide - Identifies the original author's personal guidelines for incorporating Copyright and License notices in Command Line Interface(s) and Graphical-style User Interface(s).
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> <ul style="list-style-type: none"> 1 SCOPE (System Specification) 2 DEFINITIONS, ABBREVIATIONS, AND ACRONYMS 3 OVERVIEW OF REQUIRED WORK <ul style="list-style-type: none"> a) Purpose b) Requirements and Constraints c) Concept 4 PLANS FOR PERFORMING GENERAL SOFTWARE DEVELOPMENT ACTIVITIES <ul style="list-style-type: none"> a) Software Development Process 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

		<ul style="list-style-type: none"> 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 <p>6 SCHEDULES AND ACTIVITY NETWORK</p> <p>7 PROJECT ORGANIZATION AND RESOURCES</p> <ul style="list-style-type: none"> a) Project organization b) Project Resources <p>8 NOTES</p> <p>9 APPENDIXES</p> <p>10 TO-DO-LIST</p> <ul style="list-style-type: none"> a) New Features b) Modifications c) Troubleshooting d) Validation/Regression Test <p>11 SAMPLE SCREEN SHOTS</p>
5	System Specification	<p>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. of the software system).</p> <p>Included are the following topics:</p> <ul style="list-style-type: none"> 1 SCOPE (System Specification) 2 DEFINITIONS, ABBREVIATIONS, AND ACRONYMS 3 REQUIREMENTS 4 QUALIFICATION PROVISIONS a) Demonstration

		<ul style="list-style-type: none"> b) Test c) Analysis d) Inspection e) Special Qualification Methods
		5 REQUIREMENTS TRACEABILITY
		6 NOTES
		<ul 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
		18 APPENDIX K - SYSTEM QUALITY FACTORS
		19 APPENDIX L - DESIGN AND CONSTRUCTION CONSTRAINTS
		20 APPENDIX M - PERSONNEL-RELATED REQUIREMENTS
		21 APPENDIX N - TRAINING-RELATED REQUIREMENTS
		22 APPENDIX O - LOGISTICS-RELATED REQUIREMENTS

		<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> <p>a) Required States and Modes</p> <p>b) CSCI Capability Requirements</p> <p>c) CSCI External Interface Requirements</p> <p>d) CSCI Internal Interface Requirements</p> <p>e) CSCI Internal Data Requirements</p> <p>f) Adaptation Requirements</p> <p>g) Safety Requirements</p>

		<ul style="list-style-type: none"> 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 is fixed or an enhancement is made to the product.</p> <p>Included are the following topics:</p> <ul style="list-style-type: none"> 1 SCOPE (System Specification) 2 DEFINITIONS, ABBREVIATIONS, AND ACRONYMS 3 SYSTEM REQUIREMENTS <ul style="list-style-type: none"> a) Platform Configurations (Release Notes) b) Network Configurations (Release Notes)

		<p>4 PACKAGE CONTENTS</p> <ul style="list-style-type: none"> a) Command Line Interface Library b) Graphical Text User Interface Library <p>5 FIXED BUGS & ISSUES</p> <p>6 KNOWN BUGS & ISSUES</p> <p>7 NEW FEATURES</p> <p>8 APPLICATION NOTES</p> <ul style="list-style-type: none"> a) Installation Procedure b) User Interface Design c) Developer <p>9 PERFORMANCE TUNING</p> <p>10 ACCEPTANCE TESTING</p> <p>11 COMPONENT STATUS</p> <p>12 APPENDIXES</p> <ul 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> <p>1 SCOPE (System Specification)</p> <p>2 SOFTWARE SUMMARY</p> <ul style="list-style-type: none"> a) Software Application b) Software Inventory c) Software Environment 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 <p>2 Software Development Plan</p> <p>3 System Specification</p> <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

		<ul style="list-style-type: none"> 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 <p>4 Interface Requirements Specification</p> <p>5 Software Requirements Specification</p> <ul style="list-style-type: none"> a) APPENDIX A - Nature of System and Software <p>6 Release Notes</p> <ul style="list-style-type: none"> a) APPENDIX A - Baseline Host Computer Platforms b) APPENDIX B - API Relationship c) APPENDIX C - Deliverables <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	A list of planned development for the code.

13	Use Cases	<p>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.</p> <p>1 Computer User Use Case(s)</p> <ul style="list-style-type: none">a) Role-Specificb) System-Specificc) Application-Specific <p>2 Computer Source Code Use Case(s)</p> <ul style="list-style-type: none">a) Comparison with "wxPython"b) High, Low and Extended API Relationships
----	------------------	--

Draft

Draft

2 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 its 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 qualification test components for a Python-based Command Line Interface.
tsTestsGUI	The identifier for a library of qualification 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>
Use Case	<p>Excerpt From Wikipedia, the free encyclopedia</p> <p>"In software and systems engineering, a use case is a list of action or event steps, typically defining the interactions between a role (known in the Unified Modeling Language 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 the Systems Modeling Language (SysML) or as contractual statements.</p> <p>Use case analysis is an important and valuable requirement analysis technique that has been widely used in modern software engineering since their formal introduction by Ivar Jacobson in 1992. Use case driven development is a key characteristic of many process models and frameworks such as ICONIX, the Unified Process (UP), the IBM Rational Unified Process (RUP), and the Oracle Unified Method (OUM). With its inherent iterative, incremental and evolutionary nature, use case also fits well for agile development."</p>

TERM	DEFINITION
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 ▪ UnixUtils ▪ GnuWin32 ▪ GNUWin II ▪ MinGW ▪ LBW: Linux Binaries on Windows requires Interix to be installed first."
VMware Fusion	<p>From Wikipedia, the free encyclopedia</p> <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</p>

TERM	DEFINITION
	<p>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>
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</p>

TERM	DEFINITION
	Character Set feature (e.g., Multinational Character Set) and support for 8-bit downloadable character sets."
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) ▪ Host tools, such as wxEmulator <p>Platforms supported are : x86 and ARM"</p>
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	From Wikipedia, the free encyclopedia:

TERM	DEFINITION
	<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 Vandevor, 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 appearance of spurious lines streaking across the display.

TERM	TEST TYPE DEFINITION
Acceptance Testing	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html</p> <p>Testing the system with the intent of confirming readiness of the product and customer acceptance.</p>
Ad Hoc Testing	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html</p> <p>Testing without a formal test plan or outside of a test plan. With some projects this type of testing is carried out as an adjunct to formal testing. If carried out by a skilled tester, it can often find problems that are not caught in regular testing. Sometimes, if testing occurs very late in the development cycle, this will be the only kind of testing that can be performed. Sometimes ad hoc testing is referred to as exploratory testing.</p>
Alpha Testing	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html</p> <p>Testing after code is mostly complete or contains most of the functionality and prior to users being involved. Sometimes a select group of users are involved. More often this testing will be performed in-house or by an outside testing firm in close cooperation with the software engineering department.</p>

TERM	TEST TYPE DEFINITION
Automated Testing	Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html Software testing that utilizes a variety of tools to automate the testing process and when the importance of having a person manually testing is diminished. Automated testing still requires a skilled quality assurance professional with knowledge of the automation tool and the software being tested to set up the tests.
Beta Testing	Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html Testing after the product is code complete. Betas are often widely distributed or even distributed to the public at large in hopes that they will buy the final product when it is released.
Black Box Testing	Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html Testing software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as a specification or requirements document..
Compatibility Testing	Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html Testing used to determine whether other system software components such as browsers, utilities, and competing software will conflict with the software being tested.
Configuration Testing	Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html Testing to determine how well the product works with a broad range of hardware/peripheral equipment configurations as well as on different operating systems and software.
Functional Test	Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html Functional test can be defined as testing two or more modules together with the intent of finding defects, demonstrating that defects are not present, verifying that the module performs its intended functions as stated in the specification and establishing confidence that a program does what it is supposed to do.
Independent Verification and Validation (IV&V)	Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html The process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and doesn't fail in an unacceptable manner. The individual or group doing this work is not part of the group or organization that developed the software. A term often applied to government work or where the government regulates the products, as in medical devices.
Installation Testing	Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html Testing with the intent of determining if the product will install on a variety of platforms and how easily it installs.
Integration Testing	Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html Testing two or more modules or functions together with the intent of finding interface defects between the modules or functions. Testing completed at as a part of unit or functional testing, and sometimes, becomes its own standalone test phase. On a larger level, integration testing can involve a putting together of groups of modules and functions with the goal of completing and verifying that the system meets the system requirements. (see system testing)
Load Testing	Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html Testing with the intent of determining how well the product handles competition for system resources. The competition may come in the form of network traffic, CPU utilization or memory allocation.

TERM	TEST TYPE DEFINITION
Performance Testing	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html</p> <p>Testing with the intent of determining how quickly a product handles a variety of events. Automated test tools geared specifically to test and fine-tune performance are used most often for this type of testing.</p>
Pilot Testing	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html</p> <p>Testing that involves the users just before actual release to ensure that users become familiar with the release contents and ultimately accept it. Often is considered a Move-to-Production activity for ERP releases or a beta test for commercial products. Typically involves many users, is conducted over a short period of time and is tightly controlled. (see beta testing)</p>
Regression Testing	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html</p> <p>Testing with the intent of determining if bug fixes have been successful and have not created any new problems. Also, this type of testing is done to ensure that no degradation of baseline functionality has occurred.</p>
Security Testing	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html</p> <p>Testing of database and network software in order to keep company data and resources secure from mistaken/accidental users, hackers, and other malevolent attackers.</p>
Software Testing	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html</p> <p>The process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and doesn't fail in an unacceptable manner. The organization and management of individuals or groups doing this work is not relevant. This term is often applied to commercial products such as internet applications. (contrast with independent verification and validation)</p>
Stress Testing	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html</p> <p>Testing with the intent of determining how well a product performs when a load is placed on the system resources that nears and then exceeds capacity.</p>
System Integration Testing	<p>Testing a specific hardware/software installation. This is typically performed on a COTS (commercial off the shelf) system or any other system comprised of disparate parts where custom configurations and/or unique installations are the norm.</p>
System Test	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html</p> <p>Several modules constitute a project. If the project is long-term project, several developers write the modules. Once all the modules are integrated, several errors may arise. The testing done at this stage is called system test.</p> <p>System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.</p> <p>Testing a specific hardware/software installation. This is typically performed on a COTS (commercial off the shelf) system or any other system comprised of disparate parts where custom configurations and/or unique installations are the norm.</p>

TERM	TEST TYPE DEFINITION
Unit Test	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html</p> <p>The first test in the development process is the unit test. The source code is normally divided into modules, which in turn are divided into smaller units called units. These units have specific behavior. The test done on these units of code is called unit test. Unit test depends upon the language on which the project is developed. Unit tests ensure that each unique path of the project performs accurately to the documented specifications and contains clearly defined inputs and expected results.</p>
Unit Testing	<p>Testing of individual hardware or software units or groups of related units</p> <p>-- <i>Institute of Electrical and Electronics Engineers. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. New York, NY: 1990.</i></p>
Unit Testing (Unit)	<p>A "unit" (as defined in the software testing literature) is the smallest piece of code software that can be tested in isolation.</p> <ol style="list-style-type: none"> 1 "In isolation" means separate and apart from the application, and all other units. 2 The usual connotations associated with a unit are that <ol style="list-style-type: none"> a) It is a compilation unit recognized by the compiler b) It occupies a single file c) It is small (in "lines of code" sense) d) It is atomic, i.e., you normally do not think of breaking the unit into smaller pieces 3 Units, in object-oriented software, are classes, metaclasses, parameterized classes, and their corresponding instances. In programming languages that allow them, stand-alone procedures and functions are also units.
Unit Testing (Isolation)	<p>Unit testing (as defined in the software testing literature) requires that we test the unit in isolation. That is, we want to be able to say, to a very high degree of confidence, that any actual results obtained from the execution of test cases are purely the result of the unit under test. The introduction of other units may color our results.</p> <p>Since we can seldom, if ever, test a unit without the presence of at least some other software, unit testing involves such things as "drivers" and "stubs."</p> <p>Traditionally, a driver is a piece of software that controls (drives) the unit being tested. Drivers are usually thought of as invoking, or at least containing, the unit being tested, i.e., units (being tested) are subordinate to their respective drivers.</p> <p>Sometimes, units require the presence of other subordinate (to the unit) software. Traditionally, a stub is a piece of software that both mimics the characteristics of, and is (hopefully much) simpler than, a necessary piece of software that is immediately subordinate to the unit being tested.</p>

TERM	TEST TYPE DEFINITION
Unit Testing (Driver)	<p>Drivers are programs or tools that allow a tester to exercise/ examine in a controlling manner the unit of software being tested. A driver is usually expected to provide the following:</p> <ul style="list-style-type: none"> ▪ a means of defining, declaring, or otherwise creating, any variables, constants, or other items needed in the testing of the unit, and a means of monitoring the states of these items, ▪ any input and output mechanisms needed in the testing of the unit, ▪ a means of controlling the unit being tested, e.g., deciding when the unit will be invoked, and what information will be supplied upon invocation, ▪ the generation and/or handling of any necessary interrupts, and ▪ the generation and/or handling of any necessary exceptions.
Unit Testing (Stubs)	<p>Stubs are program units that are stand-ins for the other (more complex) program units that are directly referenced by the unit being tested. Stubs are usually expected to provide the following:</p> <ul style="list-style-type: none"> ▪ an interface that is identical to the interface that will be provided by the actual program unit, and ▪ the minimum acceptable behavior expected of the actual program unit. (This can be as simple as a return statement.) <p>The goal is to keep both drivers and stubs at a minimum level of complexity. If a driver or stub becomes too complex:</p> <ul style="list-style-type: none"> ▪ it will have to be formally tested itself, and ▪ the risk of the driver or stub masking, or contributing to, an error increases to an unacceptable level. <p>The simplest driver is indeed more complex than the simplest stub. However, drivers tend to reach a (manageable) maximum level of complexity and remain there, whereas it is not uncommon to see the complexity of (at least some) stubs come very close to the complexity of the units that they are supposed to be representing.</p> <p>Lastly, there are tools (e.g., test bed generators and test harnesses) that can automate the generation of drivers and stubs.</p>
User Acceptance Testing	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html See Acceptance Testing.</p>
White Box Testing	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html Testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose.</p>

Draft

3 OVERVIEW OF REQUIRED WORK

This section shall be divided into paragraphs as needed to establish the context for the planning described in later sections. It shall include, as applicable, an overview of:

- a. Requirements and constraints on the system and software to be developed
 - b. Requirements and constraints on project documentation
 - c. Position of the project in the system life cycle
 - d. The selected program/acquisition strategy or any requirements or constraints on it
 - e. Requirements and constraints on project schedules and resources
 - f. Other requirements and constraints, such as on project security, privacy, methods, standards, interdependencies in hardware and software development, etc.
-

The following topics highlight the nature of the work:

- **Requirements and Constraints** (see "**Requirements and Constraints (Development Plan)**" on page 52)
 - **Goals & Capabilities** (see "**Goals & Capabilities (Development Plan)**" on page 53)
 - **Non-Goals & Limitations** (see "**Non-Goals & Limitations (Development Plan)**" on page 82)
- **Concept** (on page 86)
 - **Technologies** (on page 92)
 - **Design** (on page 94)
 - **Architecture** (on page 111)

3.1 Purpose

The *TeamSTARS* "tsWxGTUI_PyVx" Toolkit provides a collection of building-block components, tests, tools and utilities for creating, enhancing, troubleshooting, maintaining and supporting application programs that are suitable for embedded systems.

1 Application Programs

Automation, communication, control, diagnostic, instrumentation and simulation application programs typically require an "operator-friendly" Command Line Interface (CLI) or a Graphical-style User Interface (GUI) that can be controlled locally or remotely.

2 Embedded Systems

Systems for commercial, industrial, medical and military applications are typically customized and optimized for a specific use. Unlike their general-purpose desktop, laptop and workstation counterparts, embedded systems typically have limited, application-specific processing, memory, communication, input/output and file storage resources. Some may have character-mode hardware only suitable for their operating system's command line console.

Footnotes

1) It is still a work-in-progress. The implementation still retains historical annotations (boiler plate and comments) to inform or remind prospective adopters, developers, enhancers, maintainers and troubleshooters to the system-wide context in which each building-block component, test, tool and utility is expected to operate.

2) The "tsStripComments" tool is provided to prepare site-package installable copies of all but the top-most application, tool and test modules. The commentless copies are human readable and retain the order and indentation (look and feel) of the annotated original.

3) The top-most application, tool and test modules must provide a statement-of-purpose that survives the "tsStripComments" process. Explicitly using "`__doc__ = ''`" statement-of-purpose `''` instead of expecting the first doc string `''` statement-of-purpose `''` to implicitly set the value of "`__doc__`".

3.2 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 53) - Describes the Functional & Interface Requirements
- **Non-Goals & Limitations (Development Plan)** (on page 82) - Describes the Design Constraints
- **Comparison of wxPython with tsWxGTUI (Development Plan)** (on page 83) - 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.

3.2.1 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 23)), 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 59)) 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 64)) 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* (see "*Developer-Sandbox*" *Directory and Import Guidelines*" on page 68) 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* (see "*Site-Package*" *Directory, Import and Installation Guidelines*" on page 74)) 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.

3.2.1.1 Release Variants

Source Code components from all Python version-specific Site-Package and Developer-Sandbox variants provide applications with the same functionality and API.

However, there are internal differences based on the Python language version and based on their role in either Toolkit building block development or in Toolkit user application development. Consequently, the components are NOT designed to be intermixed and should NOT be blindly copied from one Python version or variant to another.

Each release of the TeamSTARS "tsWxGTUI_PyVx" Toolkit includes the following application-specific variants:

1 DEVELOPER_SANDBOXES

Each sandbox variant uses a multi-layered hierarchical packaging organization and import mechanism. Its import mechanism uses programmer defined package hierarchy relationships and dynamic path generation to facilitate development activities and isolate them from interfering with previously installed products.

Components of the sandbox are organized into first-level package subdirectories ("tsLibCLI", "tsLibGUI" and "tsToolsCLI" etc.) subdirectories based on their common functional relationship. This permits packages to be imported once and to share or substitute components when appropriate. Similarly, second-level package component (building block such as "tsLoggerPkg", tool and utility module) source code is typically organized into "src" and "test" package subdirectories based on their application or quality assurance roles.

The hierarchical directory components are inter-connected via a set of "__init__.py" modules. At times, this complexity makes troubleshooting more difficult.

Each user application or Toolkit building-block module must import Toolkit modules via its explicit sandbox path (such as "from tsLibCLI import tsLogger" or "import tsLibCLI; import tsLogger").

Typically, each Python language generation has its own sandbox:

- a) The *TeamSTARS* "tsWxGTUI_Py2x" Toolkit is available for use with the second generation Python programming language , Python 2.0.0 - 2.7.10.
- b) The *TeamSTARS* "tsWxGTUI_Py3x" Toolkit is available for use with the third generation Python programming language , Python 3.0.0 - 3.4.3.

2 SITE-PACKAGES

Each site-package variant uses a single-layered packaging organization with subdirectories ("tsLibCLI", "tsLibGUI" and "tsToolsCLI" etc.) based on their common functional relationship. The components within their container package are connected via an empty "__init__.py" module.

Each user application or Toolkit building-block module must import Toolkit modules via its explicit site-package path (such as "from tsWxGTUI_Py2x.tsLibCLI import tsLogger" or "from tsWxGTUI_Py3x.tsLibCLI import tsLogger")

Typically, each Python language generation has its own site-package:

- a) The *Team*STARS "tsWxGTUI_Py2x" Toolkit is available for use with the second generation Python programming language , Python 2.0.0 - 2.7.9.
- b) The *Team*STARS "tsWxGTUI_Py3x" Toolkit is available for use with the third generation Python programming language , Python 3.0.0 - 3.4.3.

Draft

3.2.1.2 Software Life Cycle Stage

The *TeamSTARS* "tsWxGTUI_PyVx" Toolkit, like other computer software products, is evolving through a series of life cycle stages. It is currently in its pre-alpha stage at which its developers can demonstrate the look and feel of its Command Line Interface and Graphical-style User Interface (the character-mode emulation of the pixel-mode "wxPython").

Stage	Excerpts from Wikipedia, the free encyclopedia (see: <i>Software Release Life Cycle</i> (http://en.wikipedia.org/wiki/Software_release_life_cycle))
Pre-Alpha	Pre-alpha refers to all activities performed during the software project before testing. These activities can include requirements analysis, software design, software development, and unit testing. In typical open source development, there are several types of pre-alpha versions. Milestone versions include specific sets of functions and are released as soon as the functionality is complete.
Alpha	<p>The alpha phase of the release life cycle is the first phase to begin software testing (alpha is the first letter of the Greek alphabet, used as the number 1). In this phase, developers generally test the software using white-box techniques. Additional validation is then performed using black-box or gray-box techniques, by another testing team. Moving to black-box testing inside the organization is known as alpha release.[2]</p> <p>Alpha software can be unstable and could cause crashes or data loss. External availability of alpha software is uncommon in proprietary software. However, open source software, in particular, often have publicly available alpha versions, often distributed as the raw source code of the software. The alpha phase usually ends with a feature freeze, indicating that no more features will be added to the software. At this time, the software is said to be feature complete."</p>
Beta	<p>Beta, named after the second letter of the Greek alphabet, is the software development phase following alpha. It generally begins when the software is feature complete. Software in the beta phase will generally have many more bugs in it than completed software, as well as speed/performance issues and may still cause crashes or data loss. The focus of beta testing is reducing impacts to users, often incorporating usability testing. The process of delivering a beta version to the users is called beta release and this is typically the first time that the software is available outside of the organization that developed it.</p> <p>The users of a beta version are called beta testers. They are usually customers or prospective customers of the organization that develops the software, willing to test the software without charge, often receiving the final software free of charge or for a reduced price. Beta version software is often useful for demonstrations and previews within an organization and to prospective customers. Some developers refer to this stage as a preview, prototype, technical preview / technology preview (TP), or early access. Some software is kept in perpetual beta—where new features and functionality are continually added to the software without establishing a firm "final" release.</p>

Release Candidate	<p>A release candidate (RC) is a beta version with potential to be a final product, which is ready to release unless significant bugs emerge. In this stage of product stabilization, all product features have been designed, coded and tested through one or more beta cycles with no known showstopper-class bug. A release is called code complete when the development team agrees that no entirely new source code will be added to this release. There could still be source code changes to fix defects, changes to documentation and data files, and peripheral code for test cases or utilities. Beta testers, if privately selected, will often be credited for using the release candidate as though it were a finished product. Beta testing is conducted in a client's or customer's location and to test the software from a user's perspective.</p>
Release to Manufacturing	<p>The term "release to manufacturing", also known as "going gold", is a term used when a software product is ready to be delivered or provided to the customer. This build may be digitally signed, allowing the end user to verify the integrity and authenticity of the software purchase. A copy of the RTM build known as the "gold master" or GM is sent for mass duplication. RTM precedes general availability (GA), when the product is released to the public.</p> <p>It is typically used in certain retail mass-production software contexts—as opposed to a specialized software production or project in a commercial or government production and distribution—where the software is sold as part of a bundle in a related computer hardware sale and typically where the software and related hardware is ultimately to be available and sold on mass/public basis at retail stores to indicate that the software has met a defined quality level and is ready for mass retail distribution. RTM could also mean in other contexts that the software has been delivered or released to a client or customer for installation or distribution to the related hardware end user computers or machines. The term does not define the delivery mechanism or volume; it only states that the quality is sufficient for mass distribution. The deliverable from the engineering organization is frequently in the form of a golden master media used for duplication or to produce the image for the web.</p>
General Availability	<p>General availability (GA) is the marketing stage at which all necessary commercialization activities have been completed and a software product is available for purchase, depending, however, on language, region, electronic vs. media availability.[8]</p> <p>Commercialization activities could include security and compliance tests, as well as localization and world wide availability. The time between RTM and GA can be from a week to months in some cases before a generally available release can be declared because of the time needed to complete all commercialization activities required by GA. At this stage, the software has "gone live".</p>

3.2.1.3 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</p>

			<p>"__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 or Module License notice	(1) For an example see "./tsDemoArchive/tsTestsLibCLI/test_tsCommandLineEnv.py"
buildCredits	__credits__	Custom formatted Python text string identifies Package or Module Third-party Author(s)	(1) For an example see "./tsDemoArchive/tsTestsLibCLI/test_tsCommandLineEnv.py"
	__line1__	Custom formatted Python text string formats Package or Module Title, Version and Modified Date header line(s) for	(1) For an example see "./tsDemoArchive/tsTestsLibCLI/test_tsCommandLineEnv.py"

		display at run time	
	<code>__line2__</code>	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_tsComma ndLineEnv.py"</code>
	<code>__line3__</code>	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_tsComma ndLineEnv.py"</code>
	<code>__line4__</code>	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_tsComma ndLineEnv.py"</code>
buildHeader	<code>__header__</code>	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_tsComma ndLineEnv.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_tsComma ndLineEnv.py"</code>
enableDefaultComma ndLineParser	Not Applicable	Python boolean value of "True" or "False".	(1) For an example see <code>"/tsDemoArchive/tsTestsLibCLI/test_tsComma ndLineEnv.py"</code> (2) Use "True" to enable use of the one and only operator settings parser in "tsLibCLI" (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 <code>-h/--help</code> , <code>-a/--about</code> , <code>-v/--version</code> , <code>-d/--debug</code> , <code>-V/--Verbose</code> and any other keyword-value pair options and/or positional arguments.
guiMessageFilename	Not Applicable	Python Redirected Output file path and name.	(1) For an example see <code>"/tsDemoArchive/tsTestsLibGUI/test_tsWxMul tiFrameEnv.py"</code>

			<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 <code>"/tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py"</code></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 <code>"/tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py"</code></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.	(1) For an example see <code>"/tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py"</code>
guiTopLevelObjectId	Not Applicable	Python integer value for top-level wxPython-style GUI Frame object ID	<p>(1) For an example see <code>"/tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py"</code></p> <p>(2) Assign a specific value or use <code>wx.ID_ANY</code> to get the default, automatically assigned value</p>
guiTopLevelObjectName	Not Applicable	Python text string value for top-level wxPython-style GUI Frame object Name	(1) For an example see <code>"/tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py"</code>
guiTopLevelObjectParent	Not Applicable	Python instance of top-level wxPython-style GUI Frame object's parent or None as appropriate.	(1) For an example see <code>"/tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py"</code>
guiTopLevelObjectPosition	Not Applicable	Python instance of top-level wxPython-style GUI Frame object's position or <code>wx.DefaultPosition</code> 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 <code>wx.DefaultSize</code> as appropriate.	(1) For an example see <code>"/tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py"</code>
guiTopLevelObjectStatusBar	Not Applicable	Python instance of top-level wxPython-	(1) For an example see <code>"/tsDemoArchive/tsTestsLibGUI/test_tsWxMul</code>

		style GUI Frame object's StatusBar or None as appropriate.	tiFrameEnv.py"
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 "./tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py"
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 "./tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py"
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 "./tsDemoArchive/tsTestsLibCLI/test_tsApplication.py" (2) For the CLI Environment Wrapper example see "./tsDemoArchive/tsTestsLibCLI/test_tsCommandLineEnv.py" (3) For the GUI Environment Wrapper example see "./tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py"
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 "./tsDemoArchive/tsTestsLibCLI/test_tsApplication.py" (2) For the CLI Environment Wrapper example see "./tsDemoArchive/tsTestsLibCLI/test_tsCommandLineEnv.py" (3) For the GUI Environment Wrapper example see "./tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py"

3.2.1.4 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 59) 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()

```

3.2.1.5 Non-Conventional Multi-Project "GitHub" Repository

Unlike other "GitHub" repositories which contain a single project, the *TeamSTARS* "tsWxGTUI_PyVx" Toolkit Repository is to be organized into four project-specific collections of computer program source code files that the Toolkit recipient will need to install, operate, modify, port and re-distribute the Toolkit.

1 Site-Packages

Two of the projects are intended to be installed as a registered Python site-package (one for the mature Python 2x programming language and the other for the evolving Python 3x programming language). Local or remote applications that have imported the appropriate Python 2x or Python 3x "site-package" can be launched from any convenient directory on the associated local or remote computer system. Modifying a copy of one of these is the most direct way to port the Toolkit to a currently unsupported Python 1x, 2x or 3x platform.

- a) Python-2x ("tsWxGTUI_Py2x")
- b) Python-3x ("tsWxGTUI_Py3x")

2 Developer-Sandboxes

Two of the projects are NOT intended to be installed as a registered Python site-package (one for the mature Python 2x programming language and the other for the evolving Python 3x programming language). Local or remote Python 2x or Python 3x applications can only be launched from the associated "tsWxGTUI_Py2x" or "tsWxGTUI_Py3x" developer-sandbox directory. Modifying a copy of one of these is the least painful way to experiment with alternative architectures and algorithms.

- a) Python-2x ("tsWxGTUI_Py2x")
- b) Python-3x ("tsWxGTUI_Py3x")

The four projects are released together so that (despite their Python 2x and Python 3x implementation differences) they retain the identical Application Programming Interface (API) and look & feel of their User Interfaces (UI):

- a) Comand Line User Interface (CLI)
- b) Graphical User Interface (GUI)

3.2.1.5.1 "Developer-Sandbox" Directory and Import Guidelines

The *TeamSTARS* "tsWxGTUI_PyVx" Toolkit is released as a non-conventional "GitHub" repository containing two "Developer-Sandbox" projects (one for the mature Python 2x programming language and the other for the evolving Python 3x programming language).

Neither of the projects are intended to be installed as a registered Python "site-package". Each is intended only to be used as an isolated sandbox to faciitate experimentation with alterative architectures and algorithms but without interfering with the operation of any installed "site-package".

1 Directory Guidelines

Each developer-sandbox will contain the following source code subdirectories:

- a) tsDemoArchive --- Contains the following subdirectories (and each Contains all the associated tests and examples):
 - src
 - tsTutorialDeveloperSandbox
- b) tsLibCLI --- Contains all building block packages and modules associated with the Command Line Interface
- c) tsLibGUI --- Contains all building block packages and modules associated with the Graphical User Interface Interface
- d) tsToolsCLI --- Contains all Tool application building block packages and modules associated with the Command Line Interface
- e) tsToolsGUI --- Contains all Tool application building block packages and modules associated with the Graphical User Interface Interface
- f) tsUtilities --- Contains all command line shelll and python scripts used to administer, configure and maintain the computer platform

2 Import Guidelines

- a) Each Python 2x application and building block module will explicitly import library packages and modules from its own "tsWxGTUI_Py2x" developer-sandbox or from the associated Python 2x Global Module Index. Try-except logic will surround the import statements to detect and report import errors.
- b) Each Python 3x application and building block module will explicitly library packages and modules from its own "tsWxGTUI_Py3x" developer-sandbox or from the associated Python 3x Global Module Index. Try-except logic will surround the import statements to detect and report import errors.

3 Installation Guidelines

Unlike "Site-Packages", "Developer-Sandboxes" DO NOT need to be "installed" (via the distutils "setup.py" tool) in order to be registered as extensions to the Global Module Index for individual Python releases.

As a consequence, applications must be launched from the top-level directory which contains the collection of library packages:

- a) ./Developer-Sandboxes/Python-2x/tsWxGTUI_Py2x
- b) ./Developer-Sandboxes/Python-3x/tsWxGTUI_Py3x

3.2.1.5.1.1 "Developer-Sandbox" Example

The "tsWxGTUI" Toolkit uses a non-traditional mechanism, for "Developer-Sandboxes", 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 illustrates an explicit way to handle the package and module dependency of the GUI features on the CLI ones:

File comparison tools (such as Deltopia's DeltaWalker or the others listed at "https://en.wikipedia.org/wiki/Comparison_of_file_comparison_tools" are especially useful in highlighting differences between the "Developer-Sandbox" and "Site-Package" import implementations.

```
#!/usr/bin/env python
# "Time-stamp: <08/30/2015 5:54:39 AM rsg>"
'''
tsDeveloperSandbox_Import_Demo.py - Module to demonstrates the import
sequence for building block components of a TeamSTARS
"tsWxGTUI_PyVx" Toolkit developer_sndbox:

    1st stage (required) --- Command Line Interface (CLI)
                                Library
    2nd stage (optioal) --- Graphical-style User Interface (GUI)
                                Library
    3rd stage (launcher) --- tsCommandLineEnv or tsWxMultiFrameEnv.
'''
#####
#
# File: tsDeveloperSandbox_Import_Demo.py
#
# Purpose:
#
#     Module to demonstrates the import sequence for building
#     block components of a TeamSTARS "tsWxGTUI_PyVx" Toolkit
#     developer_sndbox:
#
#     1st stage (required) --- Command Line Interface (CLI)
#                                Library
#     2nd stage (optioal) --- Graphical-style User Interface (GUI)
#                                Library
#     3rd stage (launcher) --- tsCommandLineEnv or tsWxMultiFrameEnv.
#
# Usage (example):
#
#     python tsDeveloperSandbox_Import_Demo.py
#
# Capabilities:
#
#     None.
#
# Limitations:
#
#     None.
#
# Notes:
#
#     None.
#
# Classes:
#
```

```
#     None.
#
# Methods:
#
#     None.
#
# Modifications:
#
#     None.
#
# ToDo:
#
#     None.
#
#####

__title__        = 'tsDeveloperSandbox_Import_Demo'
__version__      = '1.0.0'
__date__        = '05/13/2015'
__authors__      = 'Richard S. Gordon'
__copyright__    = 'Copyright (c) 2015 ' + \
                  '%s.\n\t\tAll rights reserved.' % __authors__
__license__      = 'GNU General Public License, ' + \
                  'Version 3, 29 June 2007'
__credits__      = ''

__line1__ = '%s, v%s (build %s)' % (__title__, __version__, __date__)
__line2__ = 'Author(s): %s' % __authors__
__line3__ = '%s' % __copyright__

if len(__credits__) == 0:
    __line4__ = '%s' % __license__
else:
    __line4__ = '%s%s' % (__license__, __credits__)

__header__ = '\n\n%s\n\n %s\n %s\n %s\n' % (__line1__,
                                            __line2__,
                                            __line3__,
                                            __line4__)

mainTitleVersionDate = __line1__

#-----

separator = '''
-----
'''

#-----

import sys

#-----

YourApplicationName = sys.argv[0]
```

```

print('%s' % separator)

print('\n\nName=%s' % YourApplicationName)
print('\n\nPurpose=%s' % __doc__)

print('%s' % separator)

# Python-based CLI Mode
print('\n%s' % '1st stage (required) --- ' + \
      'Command Line Interface (CLI) Library')
try:

    CLImode = True
    print('\n\nCLImode=%s' % CLImode)

    import tsLibCLI
    print('\n\nntsLibCLI: %s' % dir(tsLibCLI))

    import tsExceptions as tse
    print('\n\nntse: %s' % dir(tse))

    import tsLogger as Logger
    print('\n\nLogger: %s' % dir(Logger))

    import tsOperatorSettingsParser
    print('\n\nntsOperatorSettingsParser: %s' % dir(tsOperatorSettingsParser))

    from tsDoubleLinkedList \
        import DoubleLinkedList
    print('\n\nDoubleLinkedList: %s' % dir(DoubleLinkedList))

    if CLImode:

        from tsCommandLineEnv \
            import CommandLineEnv

        print('\n\nCommandLineEnv: %s' % dir(CommandLineEnv))

except ImportError, importCode:

    CLImode = False
    print('\n\nCLImode=%s' % CLImode)

    fmt1 = '%s: ImportError ' % str(__title__)
    fmt2 = '(tsLibCLI); '
    fmt3 = 'importCode=%s' % str(importCode)
    msg = fmt1 + fmt2 + fmt3

    print(msg)

    raise tse.PROGRAM_EXCEPTION(
        tse.APPLICATION_TRAP,
        msg)

print('%s' % separator)

```

```
print('\n%s' % '2nd stage (optioal) --- ' + \
      'Graphical-style User Interface (GUI) library')

if CLImode:

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

        GUImode = True
        print('\n\nGUImode=%s' % GUImode)

        import tsLibGUI
        print('\n\ntsLibGUI: %s' % dir(tsLibGUI))

        import tsWx as wx
        print('\n\nwx: %s' % dir(wx))

        from tsWxMultiFrameEnv import MultiFrameEnv
        print('\n\nMultiFrameEnv: %s' % dir(MultiFrameEnv))

    except ImportError, importCode:

        GUImode = False
        print('\n\nGUImode=%s' % GUImode)

        fmt1 = '%s: ImportError ' % __title__
        fmt2 = '(tsLibGUI); '
        fmt3 = 'importCode=%s' % str(importCode)
        msg = fmt1 + fmt2 + fmt3

        print(msg)

        raise tse.UserInterfaceException(
            tse.CHARACTER_GRAPHICS_NOT_AVAILABLE,
            msg)

print('%s' % separator)

print('\n%s' % '3rd stage (launcher) --- ' + \
      'tsCommandLineEnv or tsWxMultiFrameEnv.')
```

3.2.1.5.2 "Site-Package" Directory, Import and Installation Guidelines

The *TeamSTARS* "tsWxGTUI_PyVx" Toolkit is released as a non-conventional "GitHub" repository containing two "Site-Package" projects:

The projects are intended to be installed as a registered Python site-package (one for the mature Python 2x programming language and the other for the evolving Python 3x programming language).

1 Directory Guidelines

Each site-package Contains the following source code subdirectories:

- a) tsDemoArchive --- Contains the following subdirectories (and each Contains all the associated tests and examples):

tsTestsLibCLI

tsTestsLibGUI
tsTestsToolsCLI
tsTestsToolsGUI
tsTestsToolsLibCLI
tsTestsToolsLibGUI

- b) tsLibCLI --- Contains all building block modules associated with the Command Line Interface
- c) tsLibGUI --- Contains all building block modules associated with the Graphical User Interface Interface
- d) tsToolsCLI --- Contains all Tool application modules associated with the Command Line Interface
- e) tsToolsGUI --- Contains all Tool application modules associated with the Graphical User Interface Interface
- f) tsToolsLibCLI --- Contains all Tool building block modules associated with the Command Line Interface
- g) tsToolsLibGUI --- Contains all Tool building block modules associated with the Graphical User Interface
- h) tsUtilities --- Contains all command line shell and python scripts used to administer, configure and maintain the computer platform

2 Import Guidelines

- a) Each Python 2x application and building block module will explicitly import modules from its own "tsWxGTUI_Py2x" site-package or from the associated Python 2x Global Module Index.
- b) Each Python 3x application and building block module will explicitly import modules from its own "tsWxGTUI_Py3x" site-package or from the associated Python 3x Global Module Index.

3 Installation Guidelines

- a) The Python 2x site-package will be installed from the path "./Site-Packages/Python-2x" by the command "python2.x.y setup.py install" where 2.x.y represents the target Python release.
- b) The Python 3x site-package will be installed from the path "./Site-Packages/Python-3x" by the command "python3.x.y setup.py install" where 3.x.y represents the target Python release.

4 Local or remote applications that have imported the appropriate Python 2x or Python 3x "site-package" can be launched from any convenient directory on the associated local or remote computer system. Modifying a copy of one of these is the most direct way to port the Toolkit to a currently unsupported Python 1x, 2x or 3x platform.

- a) Python-2x ("tsWxGTUI_Py2x")
- b) Python-3x ("tsWxGTUI_Py3x")

It is recommended that the "tsWxGTUI_PyVx" Toolkit evaluators download and unpack the compressed repository "tarball" or "zip" file but NOT use "pip" or "setup.py" to install it anywhere except from its own site-package subdirectory.

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.

3.2.1.5.2.1 "Site-Package" Example

The "tsWxGTUI" Toolkit uses a traditional mechanism, for "Site-Packages", to facilitate the import of building block library packages. The mechanism uses the empty "__init__.py" module in each package to establish the package's component modules.

Representative applications:

- 1 CLI mode: tsDemoArchive/tsTestsToolsCLI/tsPlatformQuery.py
- 2 GUI mode: tsDemoArchive/tsTestsLibGUI/test_tsWxMultiFrameEnv.py

Draft

Draft

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

File comparison tools (such as Deltopia's DeltaWalker or the others listed at "https://en.wikipedia.org/wiki/Comparison_of_file_comparison_tools") are especially useful in highlighting differences between the "Developer-Sandbox" and "Site-Package" import implementations.

```
#!/usr/bin/env python
# "Time-stamp: <08/30/2015 5:58:19 AM rsg>"
'''
tsSitePackage_Import_Demo.py - Module to demonstrates the import
sequence for building block components of a TeamSTARS
"tsWxGTUI_PyVx" Toolkit site-package:

    1st stage (required) --- Command Line Interface (CLI)
                                Library
    2nd stage (optioal) --- Graphical-style User Interface (GUI)
                                Library
    3rd stage (launcher) --- tsCommandLineEnv or tsWxMultiFrameEnv.
'''
#####
#
# File: tsSitePackage_Import_Demo.py
#
# Purpose:
#
#     Module to demonstrates the import sequence for building
#     block components of a TeamSTARS "tsWxGTUI_PyVx" Toolkit
#     site-package:
#
#     1st stage (required) --- Command Line Interface (CLI)
#                                Library
#     2nd stage (optioal) --- Graphical-style User Interface (GUI)
#                                Library
#     3rd stage (launcher) --- tsCommandLineEnv or tsWxMultiFrameEnv.
#
# Usage (example):
#
#     python tsSitePackage_Import_Demo.py
#
# Capabilities:
#
#     None.
#
# Limitations:
#
#     None.
#
# Notes:
#
#     None.
#
# Classes:
#
```



```

#     None.
#
# Methods:
#
#     None.
#
# Modifications:
#
#     None.
#
# ToDo:
#
#     None.
#
#####

__title__      = 'tsSitePackage_Import_Demo'
__version__    = '1.0.0'
__date__       = '05/13/2015'
__authors__    = 'Richard S. Gordon'
__copyright__  = 'Copyright (c) 2015 ' + \
                 '%s.\n\t\tAll rights reserved.' % __authors__
__license__    = 'GNU General Public License, ' + \
                 'Version 3, 29 June 2007'
__credits__    = ''

__line1__ = '%s, v%s (build %s)' % (__title__, __version__, __date__)
__line2__ = 'Author(s): %s' % __authors__
__line3__ = '%s' % __copyright__

if len(__credits__) == 0:
    __line4__ = '%s' % __license__
else:
    __line4__ = '%s%s' % (__license__, __credits__)

__header__ = '\n\n%s\n\n %s\n %s\n %s\n' % (__line1__,
                                             __line2__,
                                             __line3__,
                                             __line4__)

mainTitleVersionDate = __line1__

#-----

separator = '''
-----
'''

#-----

import sys

#-----

YourApplicationName = sys.argv[0]

```

```
print('%s' % separator)

print('\n\nName=%s' % YourApplicationName)
print('\n\nPurpose=%s' % __doc__)

print('%s' % separator)

# Python-based CLI Mode
print('\n%s' % '1st stage (required) --- ' + \
      'Command Line Interface (CLI) Library')
try:

    CLImode = True
    print('\n\nCLImode=%s' % CLImode)

    # from tsWxGTUI_Py2x import tsLibCLI
    # print('\n\nntsLibCLI: %s' % dir(tsLibCLI))

    from tsWxGTUI_Py2x.tsLibCLI import tsExceptions as tse
    print('\n\nntse: %s' % dir(tse))

    from tsWxGTUI_Py2x.tsLibCLI import tsLogger as Logger
    print('\n\nLogger: %s' % dir(Logger))

    from tsWxGTUI_Py2x.tsLibCLI import tsOperatorSettingsParser
    print('\n\nntsOperatorSettingsParser: %s' % dir(tsOperatorSettingsParser))

    ## from tsWxGTUI_Py2x.tsLibCLI import tsDoubleLinkedList
    ## print('\n\nntsDoubleLinkedList: %s' % dir(tsDoubleLinkedList))

    from tsWxGTUI_Py2x.tsLibCLI.tsDoubleLinkedList import DoubleLinkedList
    ## from tsDoubleLinkedList \
    ## import DoubleLinkedList
    print('\n\nDoubleLinkedList: %s' % dir(DoubleLinkedList))

    if CLImode:

        from tsWxGTUI_Py2x.tsLibCLI.tsCommandLineEnv import CommandLineEnv

        print('\n\nCommandLineEnv: %s' % dir(CommandLineEnv))

except ImportError, importCode:

    CLImode = False
    print('\n\nCLImode=%s' % CLImode)

    fmt1 = '%s: ImportError ' % str(__title__)
    fmt2 = '(tsLibCLI); '
    fmt3 = 'importCode=%s' % str(importCode)
    msg = fmt1 + fmt2 + fmt3

    print(msg)

    raise tse.PROGRAM_EXCEPTION(
        tse.APPLICATION_TRAP,
        msg)
```

```

print('%s' % separator)

print('\n%s' % '2nd stage (optioal) --- ' + \
      'Graphical-style User Interface (GUI) library')

if CLImode:

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

        GUImode = True
        print('\n\nGUImode=%s' % GUImode)

        # from tsWxGTUI_Py2x import tsLibGUI
        # print('\n\nntsLibGUI: %s' % dir(tsLibGUI))

        from tsWxGTUI_Py2x.tsLibGUI import tsWx as wx
        print('\n\nwx: %s' % dir(wx))

        from tsWxGTUI_Py2x.tsLibGUI.tsWxMultiFrameEnv import MultiFrameEnv
        print('\n\nMultiFrameEnv: %s' % dir(MultiFrameEnv))

    except ImportError, importCode:

        GUImode = False
        print('\n\nGUImode=%s' % GUImode)

        fmt1 = '%s: ImportError ' % __title__
        fmt2 = '(tsLibGUI); '
        fmt3 = 'importCode=%s' % str(importCode)
        msg = fmt1 + fmt2 + fmt3

        print(msg)

        raise tse.UserInterfaceException(
            tse.CHARACTER_GRAPHICS_NOT_AVAILABLE,
            msg)

print('%s' % separator)

print('\n%s' % '3rd stage (launcher) --- ' + \
      'tsCommandLineEnv or tsWxMultiFrameEnv.')

```

3.2.2 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.

3.2.3 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 ▪ Embedded (via KOAN's "wxEmbedded") 	Locally (via system console or xterm) and remotely (via xterm) accessible systems: <ul style="list-style-type: none"> ▪ Desktop ▪ Laptop ▪ Workstation ▪ 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 / 	<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,

Feature	"wxPython" / "wxWidgets" / "wxEmbedded"	"tsWxGTUI_PyVx"
	<p>release, mouse button press / release and single / double click with mouse position at every clocked sample.</p> <ul style="list-style-type: none"> Events also include window resizing. 	<p>mouse button press / release and single / double / triple click with mouse position ONLY available at time of button state change.</p> <ul style="list-style-type: none"> 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 commonly used subset features of the "nCurses" API. The "tsWxGTUI_PyVx" Toolkit emulates a typical Operating System or X window system specific API for Graphical User Interface.

Feature	"wxPython" / "wxWidgets" / "wxEmbedded"	"tsWxGTUI_PyVx"
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 terminal hardware and terminal emulator software). 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.

Feature	"wxPython" / "wxWidgets" / "wxEmbedded"	"tsWxGTUI_PyVx"
		<ul style="list-style-type: none">▪ 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.

3.3 Concept

The software development plan follows an iterative process:

- 1 Design, implement and test a prototype with technologies (such as the Python programming language) that can rapidly establish a baseline for the capabilities, limitations, performance and feasibility of the "tsWxGTUI_PyVx" Toolkit concept.
- 2 Progressively re-design, re-implement and re-test additional prototypes with technologies (such as the Objective C, C++ or C programming languages) that can improve the capabilities, limitations, performance and suitability of each "tsWxGTUI_PyVx" Toolkit release candidate.
- 3 Qualify each "tsWxGTUI_PyVx" Toolkit release candidate as a software product.

The "tsWxGTUI_PyVx" Toolkit concept reflects the requirements implicit and explicit in the Toolkit's purpose. The design synergistically integrates field proven technologies that have established broad popularity in their respective niches.

Its high-level, character-mode GUI-style Application Programming Interface (API) emulates a subset of the API for the "wxPython" binding to the popular C++ language based "wxWidgets" GUI Toolkit. The emulation uses Python's "curses" module to interface to the low-level, character-mode, "nCurses" GUI-toolkit, the de-facto standard for portable advanced terminal handling. It automatically converts positioning and sizing dimensions between the pixel units used by the application and the character units used by "nCurses".

Using "Python" technology, the toolkit enables applications to run, without modification, on platforms with 32-bit and 64-bit processor architectures under "Cygwin", "GNU/Linux", "Mac OS X", "Unix" and "Windows" type operating systems. It enables original "wxPython" applications to run with little, if any, change other than the elimination of graphic-mode features such as:

- bitmapped images (icons, logos, splashscreens)
- curved, dotted and slanted lines and shapes (arrows, circles, ovals, rectangles)
- fonts with application selectable width, height and type face

The "tsWxGTUI_PyVx" Toolkit facilitates the creation of "user friendly" applications:

- 1 It includes building blocks for creating a Command Line Interface used to perform:
 - launching, event dispatching and terminating of the top-level application specified Command Line Interface and Graphical User Interface module.
 - asynchronous exception handling
 - logging with date, time and severity level of errors, warnings, debug and progress messages to syslog, stderr, stdout, files and GUI Redirected Output window
- 2 It includes a wxWidgets extension module, tsWxGraphicalTextUserInterface, that is a surrogate for wxWidgets' platform specific interfaces: its keyboard, mouse and display devices and host operating system services. Using the terminal independent nCurses API, it initializes the nCurses hardware and software. It initializes the nCurses color palette and map between wxWidgets and nCurses colors. It receives, interprets and translates nCurses Clock Tick, Keyboard and Mouse Button Press/Release/Click events into the appropriate wxWidgets events. It makes the appropriate association between mouse clicks and the triggering GUI object based on the relative visibility of the overlapping objects. It shutdown input (from a keyboard and mouse) and output (to a two-dimensional character-mode display screen) and restores the terminal state to its state at application startup.
- 3 It includes top-level tiled (side-by-side) and overlapped (layered) components for creating a sophisticated Graphical-style User Interface featuring:
 - Dialog (one or more pop-up top level application windows for operator interaction; each dialog controls the layout and creation of the standard dialog buttons (help, minimize, maximize, resize and close))
 - Frame (one or more top level application windows for operator initiated tasks; each frame controls the layout and creation of the standard frame buttons (minimize, maximize, resize and close), menubars, toolbars and statusbars.)

- Redirected Output window (displays the UNIX-style "stdout" (progress) and "stderr" (error) messages output by an application task (Dialogs and Frames) via a "print" statement generated by Dialogs and Frames. The system appends date (year/month/day), time (hour:minute:second.millisecond) and severity level (debug, info, notice, warning, alert, error, critical, emergency), to the displayed messages and scroll the messages off the display as new ones are received. When the application designer wants to draw attention to special messages, the messages may be enhanced by text attribute and color markup controls (reversed or custom foreground and background colors, blink, bold, dim, normal, standout and underline special effects); each Redirected Output window controls the layout and creation of the standard Frame buttons (minimize, maximize, resize and close)). Enhanced text markup features support the use of nCurses font style and foreground/background color pair attributes.
 - A wxWidgets extension provides a TaskBar and associated buttons to change focus to the operator selected frame and dialog; each TaskBar window controls the layout and creation of the TaskBar status info (application name, task buttons, spinning activity indicator, current date and time)
- 4 It includes lower-level tiled (side-by-side) and overlapped (layered) components for creating a sophisticated Graphical-style User Interface featuring:
- Buttons (controls for one or more action initiating selection)
 - Check Boxes (controls for one or more Left & Right Aligned independent feature selecting buttons)
 - Control (base class for all non-displayable GUI objects)
 - Gauges (controls for Horizontal & Vertical bar graphs)
 - Menu Bars (row of one or more pull down menu selections)
 - Menus (column of one or more action initiating selections)
 - Panels (optionally labeled window containing other GUI objects)
 - Radio Boxes (controls for Horizontal & Vertical panel containing two or more radio buttons)
 - Radio Buttons (controls for two or more Left & Right Aligned mutually exclusive feature selecting buttons)
 - ScrollBar (controls for Horizontal and/or Vertical ScrollBarButtons, ScrollBarGauges and ScrolledText)
 - ScrollBarButton (a wxWidgets extension provides controls for Horizontal and/or Vertical buttons, that control a scrollable text window)
 - ScrollBarGauge (a wxWidgets extension provides controls for Horizontal and/or Vertical gauge, that display the position and size of the displayed text relative to the non-displayed text)
 - Scrolled (base class and template of controls for ScrolledWindow)
 - ScrolledText (a wxWidgets extension provides controls for a scrollable text window that select the columns (page and character offset from left to right) and rows (page and line offset from top to bottom) of text to be displayed within the available horizontal and vertical page area)
 - ScrolledWindow (panel with a Horizontal and/or Vertical ScrollBar and a scrollable text window)
 - SplashScreen (a wxWidgets extension displays a bitmapped image of applicable copyright and license information at application startup)
 - StaticText (panel for displaying text)

- StatusBar (panel with one or more Status Bar Panels)
 - StatusBarPanels (one or more panels for displaying a single piece of status information)
 - TextCtrl (controls for displaying text that may be optionally formatted and scrolled; a wxWidgets extension provides text markup features that support the use of nCurses font style and foreground/background color pair attributes.)
 - ToolBars and associated menus (row of one or more action initiating selections)
 - Window (base class for all displayable GUI objects; wxWidgets extensions translate between wxWidgets pixel-mode dimensions and nCurses character-mode dimensions)
- 5** It includes positioning and sizing layout utility components for creating a sophisticated Graphical-style User Interface featuring:
- BoxSizer (controls for application selectable horizontal or vertical layout that subdivides the client area of its parent window)
 - FlexGridSizer (Future)
 - GridBagSizer (Future)
 - GridSizer (controls for application selectable combination of horizontal and vertical layout that subdivides the client area of its parent window)
 - StaticBoxSizer (controls for application selectable horizontal or vertical layout that subdivides the client area of a bordered window that it initially creates)
 - WrapSizer (Future)
- 6** It includes event generating and handling components:
- Event (Class to establish the structure holding information about an event that is passed to a callback or member function. It is an abstract base class for other event classes.)
 - EventDaemon (Class to establish a scheduling mechanism to control the timing and sequencing of event processing.)
 - EventLoop (Class to query the queue of native events sent to the wxPython application and dispatch them to the appropriate wxEvtHandlers.)
 - EventLoopActivator (Class to emulate the wxPython API for non-graphical, curses-based platforms.)
 - EventQueueEntry (Class to establish a container for the triggering object and triggering event.)
 - EventTableEntry (Class to establish a container for event table information (eventType, id, lastId, func, userData).
 - EvtHandler (Class to handle events from the windowing system.)
- 7** When used with any multi-color X11 xterm type terminal emulators with electronic displays, it supports wxWidget-style combinations of 88-foreground and 88-background color names via mapping into the nCurses-style combinations of 8-foreground and 8-background color names:
- 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
- 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

- 8** When used with any non-color vt100 or vt220 type terminal emulators and electronic display, a WxWidgets extension supports wxWidget-style combinations of 88-foreground and 88-background color names via mapping into the nCurses-style combinations of 1-foreground and 1-background color names:

- Any Foreground and Background Color combination -> WHITE on BLACK
- 9 When used with any non-color X11 xterm type terminal emulators and electronic display, a wxWidgets extension supports wxWidget-style combinations of 88-foreground and 88-background color names via mapping into the nCurses-style combinations of 1-foreground and 1-background color names:
- Any Foreground and Background Color combination -> BLACK on WHITE
- 10 When used with any terminal emulator and electronic display, a wxWidgets extension supports wxWidget-style text font style via mapping into the nCurses-style text font style (NOTE: the font style and font color attribute can be derived via the bit-wise "OR" of the font style attribute with the color pair number attribute):
- DISPLAY_BLINK -> curses.A_BLINK
 - DISPLAY_BOLD -> curses.A_BOLD
 - DISPLAY_DIM -> curses.A_DIM
 - DISPLAY_NORMAL -> curses.A_NORMAL
 - DISPLAY_REVERSE -> curses.A_REVERSE
 - DISPLAY_STANDOUT -> curses.A_STANDOUT
 - DISPLAY_UNDERLINE -> curses.A_UNDERLINE
- 11 It supports terminals with the following keyboard, multi-color electronic display and optional mouse configurations:
- Linux Terminal with xterm and xterm-color emulation
 - Mac OS X iTerm and Terminal with xterm and xterm-color emulation
 - Microsoft Windows with cygwin (mouseless), cygwin xterm and cygwin mintty (xterm) emulation
- 12 Due to various unresolved platform specific terminal emulation anomalies, it does NOT currently support terminals with the following keyboard and single- or multi-color electronic display configurations:
- Linux Terminal with ansi, cygwin, vt100 and vt220 emulation
 - Mac OS X iTerm and Terminal with ansi, cygwin, vt100 and vt220 emulation
 - Microsoft Windows with with ansi, vt100 and vt220 emulation

3.3.1 Technologies

- 1 **"Python"** - A programming language and platform-specific virtual machine that interprets and executes the source code. Versions are available for major operating systems such as Windows, Linux/Unix, OS/2, Mac, Amiga, among others. There are even versions that run on .NET, the Java virtual machine, and Nokia Series 60 cell phones. The same "Python" application program source code will run unchanged across all implementations.
- 2 **"wxWidgets"** - A popular cross-platform, Graphical User Interface (GUI) Toolkit. It is implemented in the C++ programming language. It features a sophisticated, high-level API. It allows developers to target Windows 95/98/ME, Windows NT/2K/XP, Linux/Unix with the GTK+ toolkit (or plain X11, or Motif), and MacOS.

- 3 **"wxPython"** - A popular interface (binding) to "wxWidgets" used by developers working in the "Python" programming language.
- 4 **"ncurses"** - A programming library that provides a primitive, low level Application Programming Interface (API) which allows the programmer to write text 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.
- 5 **"Hypervisor"** - A hardware virtualization techniques allowing multiple operating systems, termed guests, to run concurrently on a host computer.

From Wikipedia, the free encyclopedia

"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.

The term can be used to describe the interface provided by the specific cloud computing functionality infrastructure as a service (IaaS).[1][2]

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 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]"

a) Parallels Desktop

From Wikipedia, the free encyclopedia:

"Parallels, Inc. is a privately held virtualization technology company with offices in the USA, Germany, UK, France, Japan, China, Russia, Australia and Ukraine. Parallels headquarters is in Renton, Washington. The company has more than 900 employees as of 2013.

Parallels, Inc. was an SWsoft company until January 2008; each company operated as a separate entity and maintained its own distinct branding. In December 2007, Parallels' parent company SWsoft announced its plans to change its name to Parallels and ship both companies' products under the Parallels name. The merger was formalized in January 2008. Parallels has development centers in Moscow and Novosibirsk.

Parallels Desktop for Mac by Parallels, Inc., is software providing hardware virtualization for Macintosh computers with Intel processors."

b) VMware and VMware Fusion

From Wikipedia, the free encyclopedia:

"Founded in 1998, VMware is based in Palo Alto, California. In 2004 it was acquired by and became a subsidiary of EMC Corporation, then on August 14, 2007, EMC sold 15% of the company in a New York Stock Exchange IPO. The company trades under the symbol VMW.

VMware's desktop software runs on Microsoft Windows, Linux, and Mac OS X, while its enterprise software hypervisors for servers, VMware ESX and VMware ESXi, are bare-metal hypervisors that run directly on server hardware without requiring an additional underlying operating system.

VMware Fusion is a software hypervisor developed by VMware for computers running OS X 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."

3.3.2 Design

From Wikipedia, the free encyclopedia

Agile software development is a group of software development methods based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams. It promotes adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, and encourages rapid and flexible response to change. It is a conceptual framework that promotes foreseen interactions throughout the development cycle. The Agile Manifesto[1] introduced the term in 2001.

Software design is a process of problem solving and planning for a software solution. After the purpose and specifications of software are determined, software developers will design or employ designers to develop a plan for a solution. It includes low-level component and algorithm implementation issues as well as the architectural view.

The "tsWxGTUI_PyVx" Toolkit shall provide a unique, synergistic design that integrates the proven technologies of "Python", "ncurses", "wxWidgets" and "wxPython" which have established broad popularity in their respective niches:

- 1 **"Python"** - Contributes the platform independent programming that facilitates rapid development, troubleshooting and maintenance.
- 2 **"wxWidgets"** - Contributes the platform independent Graphical User Interface that establishes the architectural, functional, interface and exception handling requirements of candidate GUI components.
- 3 **"wxPython"** - Contributes the Python specific Graphical User Interface API that establishes the programmer reference documentation for candidate GUI components.
- 4 **"ncurses"** - Contributes the terminal independent input/output that facilitates efficient communication with local and remote keyboard, mouse and character-mode display devices.

3.3.2.1 API Technology

NOTES:

1) 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.

2) High Level API - 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. For example, the programmer can invoke a high level procedure to append one or more text strings to an internal buffer.

3) 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.

This section and the following table describes the conceptual purpose, scope, capabilities, limitations and relationship between the APIs upon which the "tsWxGTUI_PyVx" Toolkit is based and dependant. It also describes those API extensions introduced by the "tsWxGTUI_PyVx" Toolkit.

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		Local & Remote Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application login to remote platform host launch remote application terminate remote application logout of remote platform host terminate local shell logout of local platform host
	Local Host Operating System Process API <ul style="list-style-type: none"> Process Launcher Process Terminator Process Cleanup Process Event Handlers 	Local & Remote Host Operating System Process API <ul style="list-style-type: none"> Process Launcher Process Terminator Process Cleanup Process Event Handlers

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		Local & Remote Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application login to remote platform host launch remote application terminate remote application logout of remote platform host terminate local shell logout of local platform host
	<ul style="list-style-type: none"> Process Input/Output Handlers 	<ul style="list-style-type: none"> Process Input/Output Handlers
		Command Line Interface Low Level API (ts) <ul style="list-style-type: none"> tsApplication (registers and validates instantiation settings input from application via caller parameter list; registers and validates operator application settings inputs from command line keyword-value pairs and positional arguments.) tsCommandLineEnv (platform configuration, initialization, input/output supervisor and application launcher) tsCommandLineInterface (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) tsCommandLineParser (uses "optparse" to parse a Command Line and extracts keyword-value pair options and/or positional arguments) tsConfig (config file reader/writer that supports nested named sections) tsDecorators (defines a general purpose decorator factory and common decorators that takes a caller function as input and returns a decorator with the same

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		Local & Remote Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application login to remote platform host launch remote application terminate remote application logout of remote platform host terminate local shell logout of local platform host
		attributes) <ul style="list-style-type: none"> tsExceptions (defines and handles error exceptions; maps run time exception types into 8-bit exit codes and prints associated diagnostic message and traceback info) tsLogger (defines and handles prioritized, time and date stamped event message formatting and output to files, syslog, stderr, stdout and stdscr (nCurses display screen); also supports "wxPython"-style logging of assert and check case results) tsOperatorSettingsParser (uses "arparse", "optparse" or "getopt" to parse a Command Line and extracts keyword-value pair options and/or positional arguments) tsPlatformRunTimeEnvironment (captures current hardware, software and network information about the run time environment for a user process) tsReportUtilities (formats 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)

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		Local & Remote Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application login to remote platform host launch remote application terminate remote application logout of remote platform host terminate local shell logout of local platform host
		<ul style="list-style-type: none"> tsSysCommands (issues shell commands to and receives responses from the host operating system) tsThreadPool (maintains a pool of worker threads to perform time consuming operations in parallel)
Graphical User Interface Launcher <ul style="list-style-type: none"> login to local platform host and its GUI Desktop launch local application from GUI Desktop terminate local application from GUI Desktop logout of local platform host and its GUI Desktop 		Local & Remote Graphical User Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application login to remote platform host launch remote application terminate remote application logout of remote platform host terminate local shell logout of local platform host
	Local Host Operating System Process API <ul style="list-style-type: none"> Process Launcher Process Terminator Process Cleanup Process Event Handlers Process Input/Output Handlers 	Local & Remote Host Operating System Process API <ul style="list-style-type: none"> Process Launcher Process Terminator Process Cleanup Process Event Handlers Process Input/Output Handlers
Graphical User Interface Application Launcher API (wxWidgets & wxPython)	Graphical User Interface Application Launcher API (nCurses) <ul style="list-style-type: none"> application run time library 	Graphical User Interface Application Launcher API (tsWxGTUI) <ul style="list-style-type: none"> tsWxMultiFrameEnv (run time

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		Local & Remote Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application login to remote platform host launch remote application terminate remote application logout of remote platform host terminate local shell logout of local platform host
<ul style="list-style-type: none"> run time library components provide platform specific configuration, initialization, input/output supervisor and application launcher 	components provide platform specific nCurses configuration, initialization, input/output supervisor and application launcher	library component provides platform independent configuration, initialization, input/output supervisor and application launcher, event handler and terminator) <ul style="list-style-type: none"> tsWxGraphicalTextUserInterface (nCurses configuration, initialization and input/output supervisor)
Graphical User Interface High Level Application Launcher API (wxWidgets & wxPython) <ul style="list-style-type: none"> PyApp (start wxPython application) PySimpleApp (start simple wxPython application) PyOnDemandOutput (start Redirected Output window) 		Graphical User Interface High Level Application Launcher API (tsWxGTUI) <ul style="list-style-type: none"> PyApp (start wxPython application) PySimpleApp (start simple wxPython application) PyOnDemandOutput (start Redirected Output window; enhancements include date, time and message severity level annotations with and without font style and foreground/background color markup attributes)
	Graphical User Interface Low Level Launcher (nCurses) <ul style="list-style-type: none"> start (curses.start) stop (curses.stop) 	Graphical User Interface Low Level Launcher API (tsWxGTUI) <ul style="list-style-type: none"> start (tsWxGTUI.start) stop (tsWxGTUI.stop)

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		Local & Remote Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application login to remote platform host launch remote application terminate remote application logout of remote platform host terminate local shell logout of local platform host
GUI (wxWidgets & wxPython) <ul style="list-style-type: none"> It is 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+. 	GUI (nCurses) <ul style="list-style-type: none"> It is a programming library that provides an API which allows the programmer to write text mode 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. <p>On-going development of the nCurses programming library has produced the following extensions:</p> <ul style="list-style-type: none"> The curses.ascii module provides utilities for working with ASCII characters, regardless of your locale settings. The curses.panel module provides a panel stack extension that adds depth to curses windows. The curses.textpad module provides an editable text widget for curses supporting Emacs-like bindings. The curses.wrapper module provides a convenience function to ensure proper terminal setup and resetting on application entry 	GUI (tsWxGTUI) <ul style="list-style-type: none"> It is a Python and nCurses based programming library that lets developers create wxWidgets and wxPython style applications for Windows, OS X, Linux and UNIX on 32-bit and 64-bit architectures. It emulates a subset of the wxWidgets and wxPython API that is suitable for text mode terminals. It requires the operator to preadjust the position, size and appearance of each command shell window, within the display, before running an application program.

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		Local & Remote Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application login to remote platform host launch remote application terminate remote application logout of remote platform host terminate local shell logout of local platform host
	and exit.	
<ul style="list-style-type: none"> It has popular language bindings for Python, Perl, Ruby and many other languages. 		
<ul style="list-style-type: none"> Unlike other cross-platform toolkits, wxWidgets gives its applications a truly native look and feel because it uses the platform's native API rather than emulating the GUI. 		
<ul style="list-style-type: none"> It's also extensive, free, open-source and mature. 		
<ul style="list-style-type: none"> It supports local terminals with various keyboard, mouse and pixel mode display device configurations. 	<ul style="list-style-type: none"> It supports local and remote terminals with various keyboard, mouse and text mode display device configurations. 	<ul style="list-style-type: none"> It supports local and remote terminals with various keyboard, mouse and text mode display device configurations.
<ul style="list-style-type: none"> It enables the operator to adjust the display position, size and appearance of the top level GUI objects. 	<ul style="list-style-type: none"> It requires the operator to preadjust the position, size and appearance of each command shell window, within the display, before running an application program. It requires the operator to login to each remote computer before running an application program. 	<ul style="list-style-type: none"> It requires the operator to login to each remote computer before running an application program.

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		Local & Remote Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application login to remote platform host launch remote application terminate remote application logout of remote platform host terminate local shell logout of local platform host
<ul style="list-style-type: none"> It enables application programmers to control the initial display position, size and appearance of the top level GUI objects. 	<ul style="list-style-type: none"> It enables application programmers to control the initial position, size and appearance of the top level GUI objects, within a command shell window. Application programs may or may not malfunction or terminate after an operator re-adjusts the size of the command shell window. For example, the Midnight Commander application from the Free Software Foundation automatically adjusts itself to changes in screen size when used with an xterm type terminal emulator but needs to be manually refreshed via Ctrl-L when used with a cygwin console shell. 	<ul style="list-style-type: none"> It enables application programmers to control the initial position, size and appearance of the top level GUI objects, within a command shell window. Application programs will malfunction or terminate after an operator re-adjusts the size of the command shell window.
Host Platform High Level Interface (Host) Platform-specific API, libraries and User Interface for starting, initializing, configuring, controlling, monitoring and terminating computer hardware and software activities. <ul style="list-style-type: none"> Linux (Ubuntu 12.04) Mac OS X (Lion 10.7.5) Windows Pro (8/7 SP1/XP SP3) with Cygwin (1.7.4), a free add-on from Red Hat that 	Host Platform Low Level Interface (Virtual Machine) Programming language specific API, platform-specific libraries and User Interface for starting, initializing, configuring, controlling, monitoring and terminating application software activities. <ul style="list-style-type: none"> Python 2.7.3 and/or Python 3.3.3 (provides platform independent Virtual Machine API) Python Curses Module (provides terminal independent keyboard, 	Host Platform Low Level Interface (tsWxGTUI analogous to host services provided by "gtk", "msw", "osx" and "unix") Programming language specific API, platform-specific libraries and User Interface for starting, initializing, configuring, controlling, monitoring and terminating application software activities. <ul style="list-style-type: none"> tsWxGraphicalTextUserInterface (implements application specific configuration, startup, splash

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		Local & Remote Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application login to remote platform host launch remote application terminate remote application logout of remote platform host terminate local shell logout of local platform host
provides the Unix-like command line interface and tools	mouse and display API) <ul style="list-style-type: none"> NCurses Library (implements platform specific keyboard, mouse and display API) 	screen Bitmap Image display, monitoring, controlling and shutdown portions of keyboard, mouse and display API)
	TopLevel Windows (wxWidgets & wxPython) <ul style="list-style-type: none"> Frame (A GUI object whose size and position can usually be changed by the user. It usually has thick borders and a title bar, and can optionally contain a menu bar, toolbar and status bar. A frame can contain any GUI object that is not a frame or dialog. Frames are windows associated with an application task (such as an internet WEB Browser) that can be minimized into an icon, expanded to full screen or terminated upon operator demand.) Dialog (A GUI object with a title bar and sometimes a system menu, which can be moved around the screen. It can contain controls and other GUI objects and is often used to allow the user to make some choice or to answer a question. Dialogs are pop-up windows associated with an application task's menu bar (such as an input field for an operator's search criteria and an output field for a list of candidate WEB sites) that will be 	Top Level Windows (tsWxGTUI implements feature(s) available in wxWidgets or wxPython library) <ul style="list-style-type: none"> Frame (A GUI object whose size and position can (usually) be changed by the user. It usually has thick borders and a title bar, and can optionally contain a menu bar, toolbar and status bar. A frame can contain any GUI object that is not a frame or dialog. Frames are windows associated with an application task (such as an internet WEB Browser) that can be minimized into an icon, expanded to full screen or terminated upon operator demand.) Dialog (A GUI object with a title bar and sometimes a system menu, which can be moved around the screen. It can contain controls and other GUI objects and is often used to allow the user to make some choice or to answer a question. Dialogs are pop-up windows associated with an application task's menu bar (such as an input field for an operator's search criteria and an output field for a list of candidate

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		Local & Remote Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application login to remote platform host launch remote application terminate remote application logout of remote platform host terminate local shell logout of local platform host
	terminated upon completion.) <ul style="list-style-type: none"> Redirected Output (An optional window to display the UNIX-style "stdout" (progress) and "stderr" (error) messages output by an application task via a "print" statement.) 	WEB sites) that will be terminated upon completion.) <ul style="list-style-type: none"> Redirected Output (An optional window to display the UNIX-style "stdout" (progress) and "stderr" (error) messages output by an application task via a "print" statement. All output is automatically prefixed with the date (year/month/day); the time (hour:minute:second.millisecond such as "2011/01/23 12:34:56.789"; a separator (" - "); an optional severity level indicator (DEBUG, WARNING, ERROR etc. When the application designer wants to draw attention to special messages, the messages may be enhanced by: Reversed or custom foreground and background colors; blink, bold, dim, normal, standout and underline special effects)
		Top Level Windows (tsWxGTUI implements feature(s) not available in wxWidgets or wxPython library) <ul style="list-style-type: none"> Redirected Output (AppendText applies Color and Font Attribute Markup) TaskBar (buttons shift focus in manner similar to native host GUI that underlies wxWidgets and wxPython)

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		Local & Remote Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application login to remote platform host launch remote application terminate remote application logout of remote platform host terminate local shell logout of local platform host
	Low Level Windows (wxWidgets & wxPython) <ul style="list-style-type: none"> Buttons (one or more action initiating selection) Check Boxes (one or more Left & Right Aligned independent feature selecting buttons) Gauges (Horizontal & Vertical bar graphs) Menu Bars (row of one or more pull down menu selections) Menus (column of one or more action initiating selections) 	<ul style="list-style-type: none"> Low Level Windows ((tsWxGTUI implements feature(s) available in wxWidgets or wxPython library) Buttons (one or more action initiating selection) Check Boxes (one or more Left & Right Aligned independent feature selecting buttons) Gauges (Horizontal & Vertical bar graphs) Menu Bars (row of one or more pull down menu selections) Menus (column of one or more action initiating selections)
	<ul style="list-style-type: none"> Panels (optionally labeled window containing other GUI objects) Radio Boxes (Horizontal & Vertical panel containing two or more radio buttons) Radio Buttons (two or more Left & Right Aligned mutually exclusive feature selecting buttons) ScrollBar (panel with Horizontal and/or Vertical buttons, that control a scrollable text window, and a gauge, to display the position and size of the displayed text relative to the non-displayed text) ScrolledWindow (panel with a Horizontal and/or Vertical 	<ul style="list-style-type: none"> Panels (optionally labeled window containing other GUI objects) Radio Boxes (Horizontal & Vertical panel containing two or more radio buttons) Radio Buttons (two or more Left & Right Aligned mutually exclusive feature selecting buttons) ScrollBar (panel with Horizontal and/or Vertical buttons, that control a scrollable text window, and a gauge, to display the position and size of the displayed text relative to the non-displayed text) ScrolledWindow (panel with a Horizontal and/or Vertical

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		Local & Remote Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application login to remote platform host launch remote application terminate remote application logout of remote platform host terminate local shell logout of local platform host
	ScrollBar and a scrollable text window)	ScrollBar and a scrollable text window)
	<ul style="list-style-type: none"> Splash Screen (Optional window to display a bitmap description of the application. It briefly appears before any top-level wxPython-style application Frames.) StaticText (panel for displaying text) Status Bars (panel with one or more Status Bar Panels) Status Bar Panels (panel for displaying a single piece of status information) TextCtrl (panel for displaying text that may be optionally formatted and scrolled) Tool Bars (row of one or more 	<ul style="list-style-type: none"> Splash Screen (Optional window to display a text-based, bitmap-like description of the application. It briefly appears before any top-level wxPython-style application Frames.) StaticText (panel for displaying text) Status Bars (panel with one or more Status Bar Panels) Status Bar Panels (panel for displaying a single piece of status information) TextCtrl (panel for displaying text that may be optionally formatted and scrolled)

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		Local & Remote Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application login to remote platform host launch remote application terminate remote application logout of remote platform host terminate local shell logout of local platform host
	action initiating selections)	<ul style="list-style-type: none"> Tool Bars (row of one or more action initiating selections)

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		Local & Remote Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application login to remote platform host launch remote application terminate remote application logout of remote platform host terminate local shell logout of local platform host
		Low Level Windows ((tsWxGTUI implements feature(s) not available in wxWidgets or wxPython library) <ul style="list-style-type: none"> Buttons (BORDER_SIMPLE for single line label replaced by bracketed label) Panels (optionally labeled window containing other GUI objects) ScrollBar Buttons (implements feature(s) not available in nCurses library) ScrollBar Gauges (implements feature(s) not available in NCurses library) Scrolled (Template for scrollable text wndows with Horizontal and/or Vertical ScrollBars) Scrolled Text (implements text markup feature(s) not available in NCurses library) ScrolledWindow (scrollable text window with Horizontal and/or Vertical ScrollBars)Task Bar Buttons (implements feature(s) not available in NCurses library to apply keyboard focus shift) TextCtrl (AppendText applies Color and Font Attribute Markup)

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		Local & Remote Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application login to remote platform host launch remote application terminate remote application logout of remote platform host terminate local shell logout of local platform host
GUI Object Styles (wxWidgets & wxPython) <ul style="list-style-type: none"> Overlapping Tiled Border None Border Simple Splash Screen (Bitmap Image) 	GUI Object Layout Sizers (wxWidgets & wxPython) <ul style="list-style-type: none"> Box (automatically scaled Horizontal & Vertical Layout) Grid (automatically scaled Horizontal & Vertical Layout) Splash Screen (Bitmap Image display) Static Box (combines Box Sizer with Bordered Panel layout) 	GUI Object Styles (tsWxGTUI) <ul style="list-style-type: none"> Box (automatically scaled Horizontal & Vertical Layout) Grid (automatically scaled Horizontal & Vertical Layout) Splash Screen (off-line, NCurses-style Bitmap Image builder utility) Static Box (combines Box Sizer with Bordered Panel layout)
Text (wxWidgets & wxPython) <ul style="list-style-type: none"> Multiple Proportional (such as Times and Helvetica) and Monospaced (Courier) Font Families Multiple Sizes (8pt to 288pt) Special Effects (numerous including horizontal, vertical, rotated etc.) 	Text (nCurses) <ul style="list-style-type: none"> Single Monospaced (configurable by terminal operator such as Courier) Font Single Size (configurable by terminal operator such as 12pt having 8 pixel width and 12 pixel height) Special Effects (configurable by application programmer but limited to one or a combination of the following: Blinking, Bold, Dim, Normal, Reverse, Standout and Underline) 	Text (tsWxGTUI) <ul style="list-style-type: none"> Single Monospaced Font Single Size (conversions between wxWidgets pixel and nCurses character dimensions presumes 12pt font having 8 pixel width and 12 pixel height) Special Effects (configuration file "tsWxGlobals" establishes defaults for various nCurses text markup styles that may be changed or overridden by the application programmer)

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		Local & Remote Command Line Interface Launcher <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application login to remote platform host launch remote application terminate remote application logout of remote platform host terminate local shell logout of local platform host
Color Palette (wxWidgets & wxPython) <ul style="list-style-type: none"> 68-colors 4624-foreground/background color pairs 	Color Palette (nCurses) <ul style="list-style-type: none"> 8-colors (black, blue, cyan, green, magenta, red, white, yellow) 64-foreground/background color pairs 256-colors (option requires wide-character build of nCurses and Python Curses modules) 65536-foreground/background color pairs (option requires wide-character build of nCurses and Python Curses modules) Non-Color Palette (nCurses) <ul style="list-style-type: none"> Platform-specific 1-color is visible for Foreground (White) and NOT visible (Black) for Background. The 1-color (2-color pair) is determined either by the phosphor (Green, Orange, White) used in the physical terminal's display or by the color adopted by the terminal emulator (Cygwin's console-based and xterm-based vt100 emulators use White on Black while Apple's Mac OS X xterm-based vt100 emulator uses Black on White) 	Color Palette (tsWxGTUI) <ul style="list-style-type: none"> tsWxGraphicalTextUserInterface maps wxWidgets 68-colors (4624-color pairs) into nCurses 8-colors (64-color pairs) tsWxGraphicalTextUserInterface defines nCurses 256-colors (65536-color pairs) into wxWidgets 68-colors (4624-color pairs) and 188 other colors (60912 other color pairs) Non-Color Palette (tsWxGTUI) <ul style="list-style-type: none"> Platform-specific 1-color is visible for Foreground (White) and NOT visible (Black) for Background. The 1-color (2-color pair) is determined either by the phosphor (Green, Orange, White) used in the physical terminal's display or by the color adopted by the terminal emulator (Cygwin's console-based and xterm-based vt100 emulators use White on Black while Apple's Mac OS X xterm-based vt100 emulator uses Black on White)

3.3.3 Architecture

There are two architectural views of the design:

- ***Stand Alone System Architecture*** (on page 112) - Description of the components of and relationship between components of an isolated system operating by itself.
- ***Stand Among System Architecture*** (on page 116) - Description of the components of and relationship between two or more networked systems operating in collaboration with each other.

3.3.3.1 "tsWxGTUI_PyVx" Toolkit 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

3.3.3.2 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").

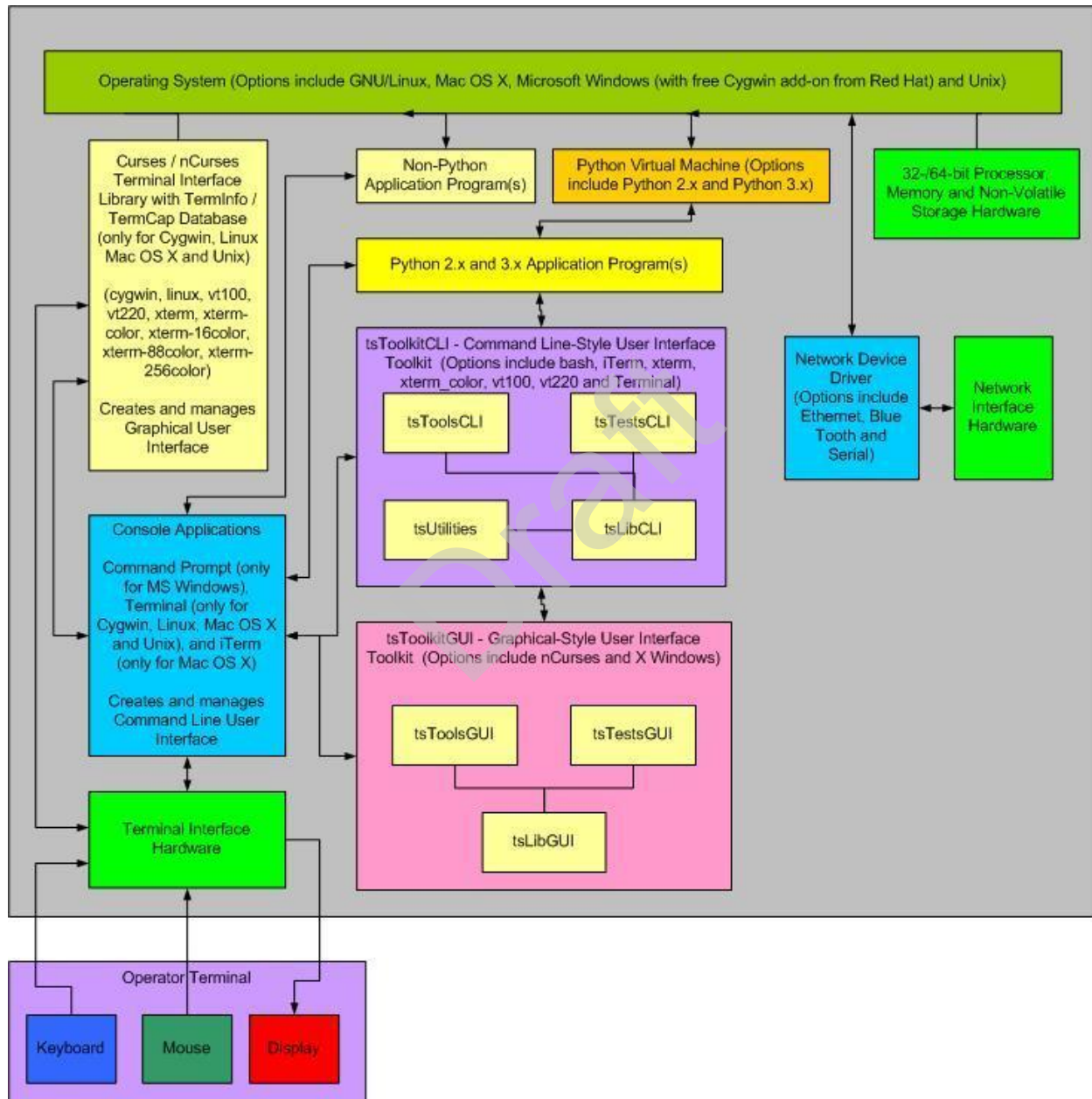
The following description uses the component names as depicted in the Block Diagram

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.

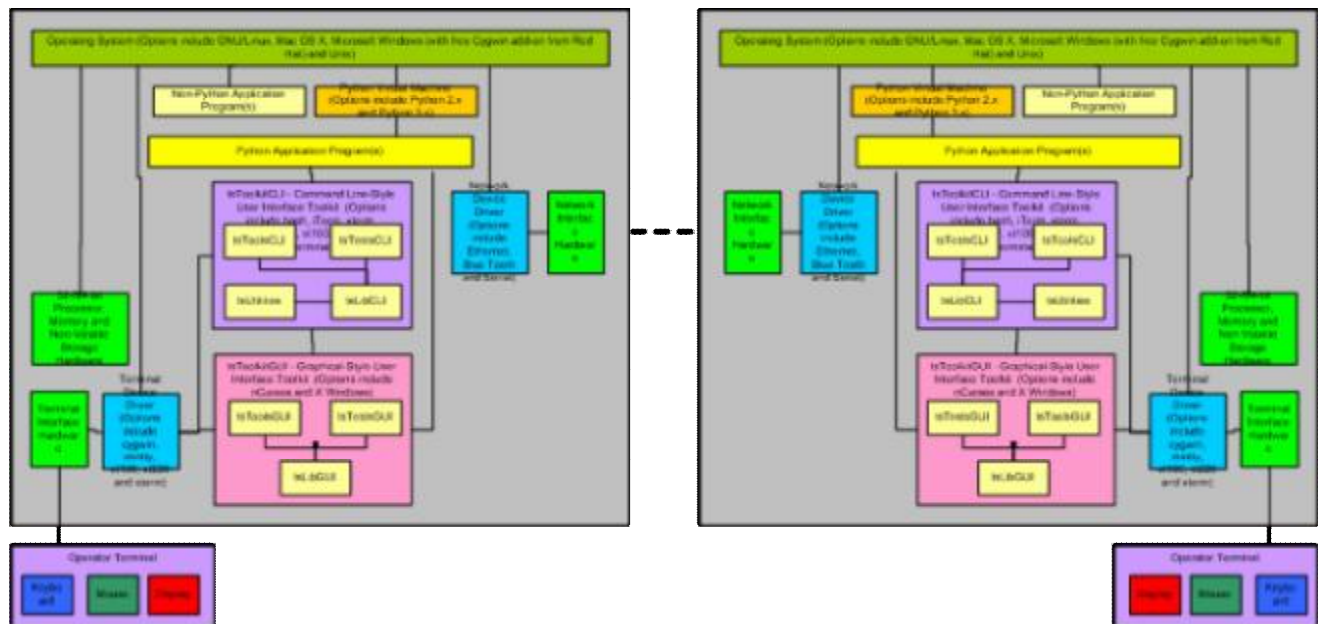
3.3.3.3 Stand Among System Architecture

The *TeamSTARS* "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

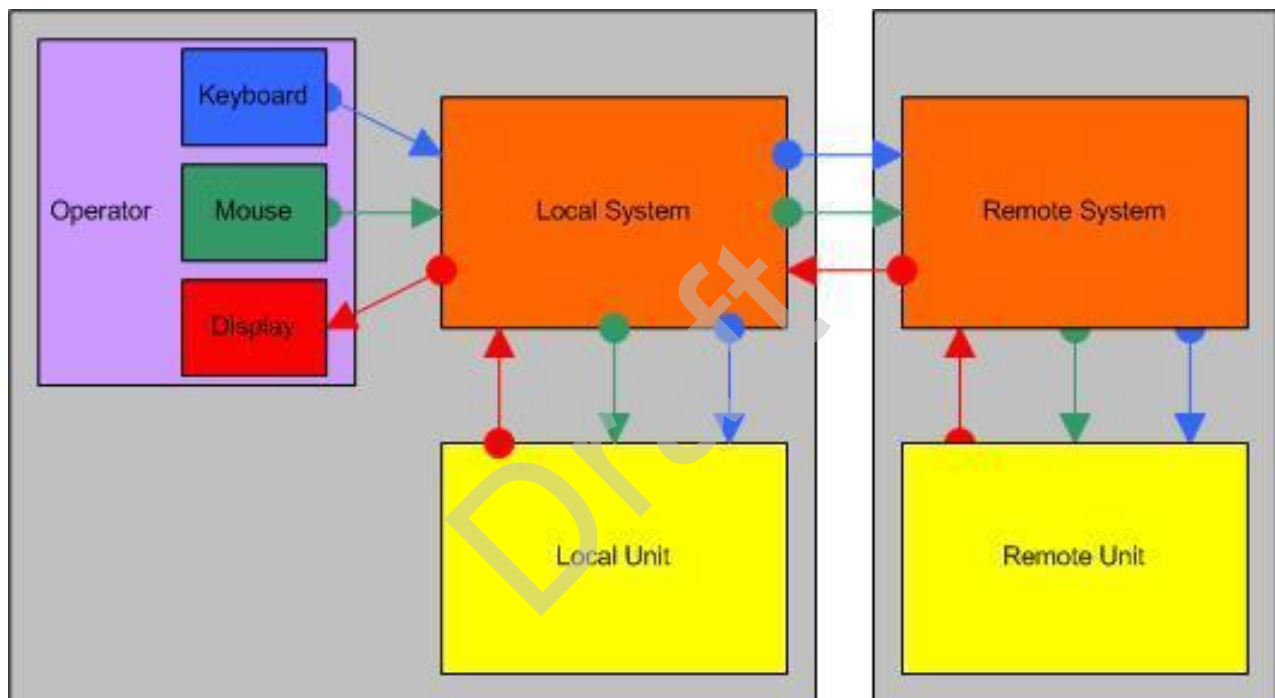
The *Stand Alone System Architecture* (on page 112), as previously described, may be extended to enable a single operator, working from a Local System, to interact with one or more Remote Systems.



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 118) 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.

3.3.3.3.1 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.

1 Application layer examples:

- a) HTTP sessions, which allow associating information with individual visitors
- b) A telnet remote login session

2 Session layer example:

- a) A Session Initiation Protocol (SIP) based Internet phone call

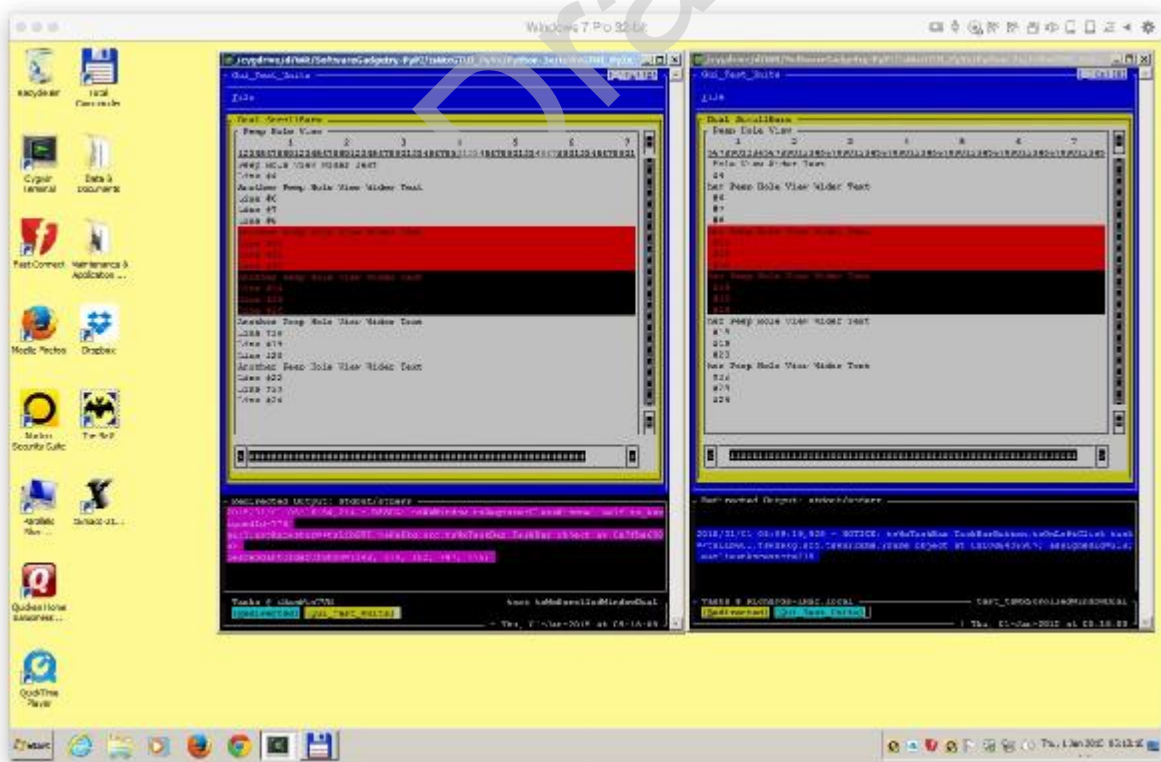
3 Transport layer example:

- a) 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).

Draft

4 PLANS FOR PERFORMING GENERAL SOFTWARE DEVELOPMENT ACTIVITIES

This section shall be divided into the following paragraphs. Provisions corresponding to non-required activities may be satisfied by the words "Not applicable." If different builds or different software on the project require different planning, these differences shall be noted in the paragraphs. In addition to the content specified below, each paragraph shall identify applicable risks/uncertainties and plans for dealing with them.

4.1 Software Development Process

This paragraph shall describe the software development process to be used. The planning shall cover all contractual clauses concerning this topic, identifying planned builds, if applicable, their objectives, and the software development activities to be performed in each build.

- 1 *Prototype* (on page 122)
- 2 *Pre-Alpha*
- 3 *Alpha* (on page 124)
- 4 *Beta* (on page 124)
- 5 *Release Candidate* (on page 125)
- 6 *Release* (on page 125)

4.1.1 Prototype

From Wikipedia, the free encyclopedia

"A prototype is an early sample, model or release of a product built to test a concept or process or to act as a thing to be replicated or learned from. It is a term used in a variety of contexts, including semantics, design, electronics, and software programming. A prototype is designed to test and trial a new design to enhance precision by system analysts and users. Prototyping serves to provide specifications for a real, working system rather than a theoretical one.[1]"

Python 2x prototypes are developed on Microsoft Windows using Cygwin, the free inux-like Command Line Interface and GNU Toolkit from Red Hat. Pylint is used to check the syntax of each Python module. WingIDE Pro is then used to debug the components.

Once Python 2x prototypes have been debugged, the Python 2to3 translation utility is used to generate the Python 3x counterparts. Pylint is used to check the syntax of each Python module. WingIDE Pro is then used to debug the components. Manual adjustments are made to correct semantic difference between the Python 2x and Python 3x versions. DeltaWalker is used to compare the Python 2x with its Python 3x counterparts.

Development chronology (reflects import dependancies):

1 tsLibCLI

- a) tsCxGlobals
- b) tsCommandLineInterfacePkg
- c) tsExceptionPkg
- d) tsReportUtilityPkg
- e) tsLoggerPkg
- f) tsDoubleLinkedListPkg
- g) tsPlatformRunTimeEnvironmentPkg
- h) tsApplicationPkg
- i) tsCommandLineEnvPkg
- j) tsOperatorSettingsPrserPkg
- k) tsSysCommandPkg

2 tsToolsCLI

- a) tsLinesOfCodeProjectMetricsPkg
- b) tsPlatformQueryPkg
- c) tsStripCommentsPkg
- d) tsStripLineNumbersPkg
- e) tsTreeCopyPkg

- f) tsTreeTrimLinesPkg
- 3** tsTestsCLI
- 4** tsUtilities
- 5** tsLibGUI
 - a) tsWxPkg
- 6** tsToolsGUI
- 7** tsTestsGUI

4.1.2 Pre-Alpha

From Wikipedia, the free encyclopedia

"Pre-alpha refers to all activities performed during the software project before testing. These activities can include requirements analysis, software design, software development, and unit testing. In typical open source development, there are several types of pre-alpha versions. Milestone versions include specific sets of functions and are released as soon as the functionality is complete."

Pre-alpha prototypes are tested on platforms having various host operating system releases (Linux, Mac OS X, Microsoft Windows and Unix) and various Python virtual machines.

- Modules may be modified to resolve latent host operating system and Python version dependencies (such as availability or absence of argparse, optparse or getopt).
- Modules may be modified to resolve latent Terminal Emulator dependencies (such as availability or absence of xterm, xterm-color, xterm_16color, xterm-88color, xterm-256color, vt100 or vt220).
- Regression tests are used to verify compliance with system specifications, functional requirements and interface requirements.

Development chronology (reflects import dependencies):

- 1** tsWxGTUI_2x-0.0.0
 - a) tsLibCLI
 - b) tsToolsCLI
 - c) tsTestsCLI
 - d) tsUtilities
 - e) tsLibGUI
 - f) tsToolsGUI
 - g) tsTestsGUI
- 2** tsWxGTUI_3x-0.0.0
 - a) tsLibCLI
 - b) tsToolsCLI
 - c) tsTestsCLI

- d) tsUtilities
- e) tsLibGUI
- f) tsToolsGUI
- g) tsTestsGUI

4.1.3 Alpha

From Wikipedia, the free encyclopedia

"The alpha phase of the release life cycle is the first phase to begin software testing (alpha is the first letter of the Greek alphabet, used as the number 1). In this phase, developers generally test the software using white box techniques. Additional validation is then performed using black box or gray box techniques, by another testing team. Moving to black box testing inside the organization is known as alpha release.[2]

Alpha software can be unstable and could cause crashes or data loss. External availability of alpha software is uncommon in proprietary software. However, open source software, in particular, often have publicly available alpha versions, often distributed as the raw source code of the software. The alpha phase usually ends with a feature freeze, indicating that no more features will be added to the software. At this time, the software is said to be feature complete."

4.1.4 Beta

From Wikipedia, the free encyclopedia

"Beta, named after the second letter of the Greek alphabet, is the software development phase following alpha. It generally begins when the software is feature complete. Software in the beta phase will generally have many more bugs in it than completed software, as well as speed/performance issues and may still cause crashes or data loss. The focus of beta testing is reducing impacts to users, often incorporating usability testing. The process of delivering a beta version to the users is called beta release and this is typically the first time that the software is available outside of the organization that developed it.

The users of a beta version are called beta testers. They are usually customers or prospective customers of the organization that develops the software, willing to test the software without charge, often receiving the final software free of charge or for a reduced price. Beta version software is often useful for demonstrations and previews within an organization and to prospective customers. Some developers refer to this stage as a preview, prototype, technical preview (TP), or early access. Some software is kept in perpetual beta—where new features and functionality are continually added to the software without establishing a firm "final" release."

4.1.5 Release Candidate

From Wikipedia, the free encyclopedia

"A release candidate (RC) is a beta version with potential to be a final product, which is ready to release unless significant bugs emerge. In this stage of product stabilization, all product features have been designed, coded and tested through one or more beta cycles with no known showstopper-class bug. A release is called code complete when the development team agrees that no entirely new source code will be added to this release. There could still be source code changes to fix defects, changes to documentation and data files, and peripheral code for test cases or utilities. Beta testers, if privately selected, will often be credited for using the release candidate as though it were a finished product. Beta testing is conducted in a client's or customer's location and to test the software from a user's perspective."

4.1.6 Release

From Wikipedia, the free encyclopedia

"Release to web (RTW) or web release is a means of software delivery that utilizes the Internet for distribution. No physical media are produced in this type of release mechanism by the manufacturer. Web releases are becoming more common as Internet usage grows."

4.2 General Plans For Software Development

This paragraph shall be divided into the following subparagraphs.

- 1 *Software Development Methods* (on page 126)
- 2 *Standards For Software Products* (on page 127)
- 3 *Reusable Software Products* (on page 147)
- 4 *Handling Of Critical Requirements* (on page 149)
- 5 *Computer Hardware Resource Utilization* (on page 150)
- 6 *Recording Rationale* (on page 150)
- 7 *Access For Acquirer Review* (on page 150)

4.2.1 Software Development Methods

This paragraph shall describe or reference the software development methods to be used. Included shall be descriptions of the manual and automated tools and procedures to be used in support of these methods. The methods shall cover all contractual clauses concerning this topic. Reference may be made to other paragraphs in this plan if the methods are better described in context with the activities to which they will be applied.

Excerpts From Wikipedia, the free encyclopedia:

"A software development methodology or system development methodology in software engineering is a framework that is used to structure, plan, and control the process of developing an information system. Common methodologies include waterfall, prototyping, iterative and incremental development, spiral development, rapid application development, and extreme programming. A methodology can also include aspects of the development environment (i.e. IDEs), model-based development, computer aided software development, and the utilization of particular frameworks (i.e. programming libraries or other tools)."

Prototyping

"Software prototyping, is the development approach of activities during software development, the creation of prototypes, i.e., incomplete versions of the software program being developed. basic principles are:[2]

- Not a standalone, complete development methodology, but rather an approach to handle selected parts of a larger, more traditional development methodology (i.e. incremental, spiral, or rapid application development (RAD)).
- Attempts to reduce inherent project risk by breaking a project into smaller segments and providing more ease-of-change during the development process.
- User is involved throughout the development process, which increases the likelihood of user acceptance of the final implementation.
- Small-scale mock-ups of the system are developed following an iterative modification process until the prototype evolves to meet the users' requirements.
- While most prototypes are developed with the expectation that they will be discarded, it is possible in some cases to evolve from prototype to working system.
- A basic understanding of the fundamental business problem is necessary to avoid solving the wrong problem."

Incremental development

"Various methods are acceptable for combining linear and iterative systems development methodologies, with the primary objective of each being to reduce inherent project risk by breaking a project into smaller segments and providing more ease-of-change during the development process. basic principles are:[2]

- A series of mini-Waterfalls are performed, where all phases of the Waterfall are completed for a small part of a system, before proceeding to the next increment, or
- Overall requirements are defined before proceeding to evolutionary, mini-Waterfall development of individual increments of a system, or

- The initial software concept, requirements analysis, and design of architecture and system core are defined via Waterfall, followed by iterative Prototyping, which culminates in installing the final prototype, a working system."

4.2.2 Standards For Software Products

This paragraph shall describe or reference the standards to be followed for representing requirements, design, code, test cases, test procedures, and test results. The standards shall cover all contractual clauses concerning this topic. Reference may be made to other paragraphs in this plan if the standards are better described in context with the activities to which they will be applied. Standards for code shall be provided for each programming language to be used. They shall include at a minimum:

- a. Standards for format (such as indentation, spacing, capitalization, and order of information)
 - b. Standards for header comments (requiring, for example, name/identifier of the code; version identification; modification history; purpose; requirements and design decisions implemented; notes on the processing (such as algorithms used, assumptions, constraints, limitations, and side effects); and notes on the data (inputs, outputs, variables, data structures, etc.)
 - c. Standards for other comments (such as required number and content expectations)
 - d. Naming conventions for variables, parameters, packages, procedures, files, etc.
 - e. Restrictions, if any, on the use of programming language constructs or features
 - f. Restrictions, if any, on the complexity of code aggregates
-

- 1 *Python Programming Standards* (on page 127)
- 2 *Shell Script Programming Standards* (on page 130)
- 3 *Document Authoring Standards* (see "*Document Authoring Standards (Draft)*" on page 132)
- 4 *Document Publication Standards* (on page 146)

4.2.2.1 Python Programming Standards

- 1 Standards for format (such as indentation, spacing, capitalization, and order of information)
Code Lay-out - To be in accordance with "**PEP 8 -- Style Guide for Python Code**"
- 2 Standards for header comments (requiring, for example, name/identifier of the code; version identification; modification history; purpose; requirements and design decisions implemented; notes on the processing (such as algorithms used, assumptions, constraints, limitations, and side effects); and notes on the data (inputs, outputs, variables, data structures, etc.)
Time-Stamp - Date, time and identification of most recent file editor.
File - Name of source file.
Purpose - Synopsis of design goals for source file's functionality and usage.
Limitations - Synopsis of design constraints on source file's functionality and usage.

Usage - Example(s) of design for importing and invoking module(s) in source file.

Classes & Methods - Summary of source file components.

Modifications - Chronological description of source file editor, changes, goals and results.

To-Do - Chronological description of future source file changes, goals and results.

3 Standards for other comments (such as required number and content expectations)

Block Comments - To be in accordance with "**PEP 8 -- Style Guide for Python Code**"

Inline Comments - To be in accordance with "**PEP 8 -- Style Guide for Python Code**"

Documentation String Comments - To be in accordance with "**PEP 8 -- Style Guide for Python Code**"

4 Naming conventions for variables, parameters, packages, procedures, files, etc.

Naming Conventions - To be in accordance with "**PEP 8 -- Style Guide for Python Code**" with following conventions to distinguish the various "tsWxGTUI_PyVx" Toolkit components from one another and from third-party ones:

a) Packages

Each "tsWxGTUI_PyVx" Toolkit package will include a "**ts**" prefix and "**Pkg**" suffix in each package name. For example, whereas the Python Global Module Index includes the "**Logging**" package, the "tsWxGTUI_PyVx" Toolkit includes the "**tsLoggerPkg**" package.

b) Building Block Libraries

Each "tsWxGTUI_PyVx" Toolkit library containing building block modules will include a "**tsLIB**" prefix in each library name. For example, the "tsWxGTUI_PyVx" Toolkit includes the "**tsLibCLI**" and "**tsLibGULI**" libraries.

c) Building Block Test Libraries

Each "tsWxGTUI_PyVx" Toolkit library containing application programs that test building block modules will include a "**tsTests**" prefix in each library name. For example, the "tsWxGTUI_PyVx" Toolkit includes the "**tsTestsCLI**" and "**tsTestsGUI**" libraries.

d) Tool Libraries

Each "tsWxGTUI_PyVx" Toolkit library containing application programs that improve developer productivity or report development project metrics will include a "**tsTools**" prefix in each library name. For example, the "tsWxGTUI_PyVx" Toolkit includes the "**tsToolsCLI**" and "**tsToolsGUI**" libraries.

e) Library Constructor Modules

Associated with each "tsWxGTUI_PyVx" Toolkit library or package is an "**__init__.py**" module.

It may be empty when it is needed only to facilitate the import operations by modules that are members of the same library or package.

It must contain special logic to facilitate the import operations by modules that are NOT members of the same library or package but are instead members of other libraries or packages in the "tsWxGTUI_PyVx" Toolkit architecture hierarchy.

f) Modules

Each "tsWxGTUI_PyVx" Toolkit module within "tsLibCLI", "tsTestsCLI" or "tsToolsCLI" will include a "**ts**" prefix in each module name.

Each "tsWxGTUI_PyVx" Toolkit module within "tsLibGUI", "tsTestsGUI" or "tsToolsGUI" will include a "**tsWx**" prefix in each module name. *NOTE: Internal logic in the "tsWx" module will conceal the prefix from those applications that use the "tsWxGTUI_PyVx" Toolkit.*

g) Classes

Each "tsWxGTUI_PyVx" Toolkit class within "tsLibCLI", "tsTestsCLI" or "tsToolsCLI" will include a "**ts**" prefix in each class name.

Each "tsWxGTUI_PyVx" Toolkit class within "tsLibGUI", "tsTestsGUI" or "tsToolsGUI" will NOT include any prefix in any character-mode emulation of any pixel-mode "wxPython" or "wxWidgets" class but may include a "**ts**" prefix in classes provided only for internal use.

h) Methods/Functions

Each "tsWxGTUI_PyVx" Toolkit method/function within "tsLibCLI", "tsTestsCLI" or "tsToolsCLI" will include a "**ts**" prefix in each method/function name.

Each "tsWxGTUI_PyVx" Toolkit method/function within "tsLibGUI", "tsTestsGUI" or "tsToolsGUI" will NOT include any prefix in any character-mode emulation of any pixel-mode "wxPython" or "wxWidgets" method/function but may include a "**ts**" prefix in a method/function provided only for internal use.

i) Variables/Constants

Each "tsWxGTUI_PyVx" Toolkit variable/constant within "tsLibCLI", "tsTestsCLI" or "tsToolsCLI" will include a "**ts_**" prefix in each variables/constant name.

Each "tsWxGTUI_PyVx" Toolkit variable/constant within "tsLibGUI", "tsTestsGUI" or "tsToolsGUI" will NOT include a "**ts_**" prefix in any character-mode emulation of any pixel-mode "wxPython" or "wxWidgets" variable/constant but may include a "**ts_**" prefix in a variable/constant provided only for internal use.

5 Restrictions, if any, on the use of programming language constructs or features

Programming Recommendations - To be in accordance with "**PEP 8 -- Style Guide for Python Code**"

6 Restrictions, if any, on the complexity of code aggregate

Programming Recommendations - Organize the software into file modules, classes, methods, functions and other logical components so as to facilitate troubleshooting, maintenance and enhancement by the original author or anyone else over a software life cycle that may extend into many decades.

Excerpt From Wikipedia, the free encyclopedia:

"Programming complexity (or software complexity) is a term that encompasses numerous properties of a piece of software, all of which affect internal interactions. According to several commentators, there is a distinction between the terms complex and complicated. Complicated implies being difficult to understand but with time and effort, ultimately knowable. Complex, on the other hand, describes the interactions between a number of entities. As the number of entities increases, the number of interactions between them would increase exponentially, and it would get to a point where it would be impossible to know and understand all of them. Similarly, higher levels of complexity in software increase the risk of unintentionally interfering with interactions and so increases the chance of introducing defects when making changes. In more extreme cases, it can make modifying the software virtually impossible. The idea of linking software complexity to the maintainability of the software has been explored extensively by Professor Manny Lehman, who developed his Laws of Software Evolution from his research. He and his co-Author Les Belady explored numerous possible Software Metrics in their oft cited book,[1] that could be used to measure the state of the software, eventually reaching the conclusion that the only practical solution would be to use one that uses deterministic complexity models."

4.2.2.2 Shell Script Programming Standards

1 Bourne Again Shell (*.bsh)

excerpt From Wikipedia, the free encyclopedia:

"Original author(s) Brian Fox

Initial release June 7, 1989; 24 years ago

Stable release 4.3 / February 27, 2014; 2 months ago[1]

Development status Active

Written in C

Operating system Cross-platform

Platform GNU

Available in English, multilingual (gettext)

Type Unix shell

License GNU GPL v3+[2]

Website www.gnu.org/software/bash/

Bash is a Unix shell written by Brian Fox for the GNU Project as a free software replacement for the Bourne shell (sh).[3][4] Released in 1989,[5] it has been distributed widely as the shell for the GNU operating system and as a default shell on Linux and Mac OS X. It has been ported to Microsoft Windows and distributed with Cygwin and MinGW, to DOS by the DJGPP project, to Novell NetWare and to Android via various terminal emulation applications.

Bash is a command processor, typically run in a text window, allowing the user to type commands which cause actions. Bash can also read commands from a file, called a script. Like all Unix shells, it supports filename wildcarding, piping, here documents, command substitution, variables and control structures for condition-testing and iteration.[6] The keywords, syntax and other basic features of the language were all copied from sh. Other features, e.g., history, were copied from csh and ksh. Bash is a POSIX shell but with a number of extensions.

The name itself is an acronym, a pun, and a description. As an acronym, it stands for Bourne-again shell, referring to its objective as a free replacement for the Bourne shell.[7] As a pun, it expressed that objective in a phrase that sounds similar to born again, a term for spiritual rebirth.[8][9] The name is also descriptive of what it did, bashing together the features of sh, csh, and ksh.[10]"

2 Bourne Shell (*.sh)

Excerpt From Wikipedia, the free encyclopedia:

"Original author(s) Stephen Bourne

Initial release 1977

Operating system Unix

Type Unix shell

License [under discussion]

The Bourne shell (sh) is a shell, or command-line interpreter, for computer operating systems.

The Bourne shell was the default Unix shell of Unix Version 7. Most Unix-like systems continue to have /bin/sh—which will be the Bourne shell, or a symbolic link or hard link to a compatible shell even when other shells are used by most users.

Developed by Stephen Bourne at Bell Labs, it was a replacement for the Thompson shell, whose executable file had the same name—sh. It was released in 1977 in the Version 7 Unix release distributed to colleges and universities. Although it is used as an interactive command interpreter, it was always intended as a scripting language and contains most of the features that are commonly considered to produce structured programs.

It gained popularity with the publication of The UNIX Programming Environment by Brian W. Kernighan and Rob Pike—the first commercially published book that presented the shell as a programming language in a tutorial form."

3 Microsoft Command Prompt Shell (*.bat)

Excerpt From Wikipedia, the free encyclopedia:

"Command Prompt A component of Microsoft Windows

Type Command-line interpreter

Included with

Windows NT

Windows CE

OS/2

Replaces COMMAND.COM

Related components

Windows PowerShell

Batch file

Command Prompt (executable name cmd.exe) is the Microsoft-supplied command-line interpreter on OS/2, Windows CE and on Windows NT-based operating systems (including Windows 2000, XP, Vista, 7, 8, Server 2003, Server 2008, Server 2008 R2 and Server 2012). It is the analog of COMMAND.COM in MS-DOS and Windows 9x systems (where it is called "MS-DOS Prompt"), or of the Unix shells used on Unix-like systems.

4.2.2.3 Document Authoring Standards (Draft)

Authoring and Publication activities shall be performed on a development platform suitably equipped with various authoring tools.

- 1 The purpose of the documentation is to establish and maintain a focus for contributors to the *TeamSTARS "tsWxGTUI_PyVx"* Toolkit development, enhancement and maintenance projects.
 - a) The model for the document set shall be based on MIL-STD-498 (See Section 6 **DEFINITIONS, ABBREVIATIONS, AND ACRONYMS** (on page 23)).
 - b) Each document topic shall include such annotations as to orient the author and reader to the topic under discussion and to maintain their focus over the course of their use of each document.
 - c) To minimize duplication of authoring, editing and reading effort, topics shall be discussed once and then referenced (with hyperlink to original document and topic) or re-used (via with identification of the document and topic containing the master) as needed.
- 2 Authoring and Publication activities shall be in accordance with the following standards and practices:
 - a) *Authoring Platform Requirements* (on page 132)
 - b) *Author-it Requirements* (on page 134)
 - c) *Microsoft Access Requirements* (on page 137)
 - d) *Microsoft Excel Requirements* (on page 138)
 - e) *Microsoft Visio Requirements* (on page 139)
 - f) *Microsoft Word Requirements* (on page 141)
 - g) *FinePrint Requirements* (on page 143)
 - h) *pdfFactory Pro Requirements* (on page 144)
 - i) *Screen Capture Requirements* (on page 145)

4.2.2.3.1 Authoring Platform Requirements

Requirements for the Authoring Platforms shall reflect the cross-platform objectives for the *TeamSTARS "tsWxGTUI_PyVx"* Toolkit:

- 1 Computer Hardware

Any 32-/64-bit general purpose computer.
- 2 Operating System Software

Any Linux, Mac OS X, Microsoft Windows or Unix.

3 Hypervisor Application Software

From Wikipedia, the free encyclopedia:

"A hypervisor or virtual machine monitor (VMM) is a piece of computer software, firmware or hardware that creates and runs virtual machines.

A computer on which a hypervisor is running one or more virtual machines is defined as a host machine. Each virtual machine is called a guest machine. The hypervisor presents the guest operating systems with 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."

a) Parallels Desktop for Mac OS X

GuestOS from various vendors include various releases of Linux, Mac OS X, Microsoft Windows and Unix

b) VMware Fusion for Mac OS X

GuestOS from various vendors include various releases of Linux, Microsoft Windows and Unix

c) VMware Workstation for Linux and Microsoft Windows

GuestOS from various vendors include various releases of Linux, Microsoft Windows and Unix

4 Authoring Content

a) Bit-mapped graphic images (screenshots with customizable content)

b) Databases (customizable key and data value based tables, queries, forms, sorting)

c) Drawings (flow and organization charts with arrows, lines, shapes and layering)

d) Spreadsheets (multi-row, multi-column and multi-sheet tables with customizable layout and cell formats)

e) Text (with customizable outline numbering, organization and detail), spreadsheet-like tables and bit-mapped graphic images.

f) Tables (with multiple rows and columns that may be split as appropriate to the application)

5 Authoring Applications

a) Applications for **Microsoft Windows** (*recommended*)

Author-it (*recommended*: Single Source Content Management)

Microsoft Office (*recommended*: Access databases, Excel spreadsheets, PowerPoint presentations, Word text processing)

Microsoft Visio (*recommended*: Flow and Organization Chart)

LibreOffice (*free alternative*: Base databases, Calc spreadsheets, Draw drawings, Impress presentations, Writer text processing)

b) Applications for **Mac OS X**

Microsoft Office (*recommended*: Excel spreadsheets, PowerPoint presentations, Word text processing)

Apple Keynote (presentations)

Apple Numbers (spreadsheets)

Apple Pages (text processing)

LibreOffice (*free alternative*: Base databases, Calc spreadsheets, Draw drawings, Impress presentations, Writer text processing)

c) Applications for **Linux**

LibreOffice (*free alternative*: Base databases, Calc spreadsheets, Draw drawings, Impress presentations, Writer text processing)

Andrew User Interface System (<http://www.cs.cmu.edu/~AUIS/andrew-home.html>) (*free alternative*: word processor, html editor, program editor, drawing editor, directory browser, help browser, spreadsheet, mail / bulletin board manager, application builder, scripting language)

d) Applications for **Unix**

Andrew User Interface System (<http://www.cs.cmu.edu/~AUIS/andrew-home.html>) (*free alternative*: word processor, html editor, program editor, drawing editor, directory browser, help browser, spreadsheet, mail / bulletin board manager, application builder, scripting language)

4.2.2.3.2 Author-it Requirements

- 1 The document masters and any derived HTML and Microsoft Word versions shall be produced using Author-it version 5 or a more recent version.

Author-it is an authoring tool and content management system for creating, maintaining, and distributing single-sourced content.

- 2 Author-it requires the following third-party services:

Excerpt from "OLE Concepts and Requirements Overview" at <http://support.microsoft.com/kb/86008>:

"SUMMARY

OLE is a technology that enables an application to create compound documents that contain information from a number of different sources. For example, a document in an OLE-enabled word processor can accept an embedded spreadsheet object. Unlike traditional "cut and paste" methods where the receiving application changes the format of the pasted information, embedded documents retain all their original properties. If the user decides to edit the embedded data, Windows activates the originating application and loads the embedded document."

- a) **Microsoft Access Requirements** (on page 137) - Author-it can import content (database tables, queries and forms) from Microsoft Access.
- b) **Microsoft Excel Requirements** (on page 138) - Author-it can import content (spreadsheets) from Microsoft Excel.
- c) **Microsoft Visio Requirements** (on page 139) - Author-it can import content (block diagrams, data flow diagrams and organizational chart drawings) from Microsoft Visio.
- d) **Microsoft Word Requirements** (on page 141) - Author-it can import content (headings, paragraphs, tables, lists and drawings) from and exports content to Microsoft Word.

- e) ***FinePrint Requirements*** (on page 143) - Author-it can "print" content to FinePrint which can optionally apply watermarks (draft, confidential etc.), format pages into a booklet format or re-size multiple (1, 2, 4 or 8) document pages to fit on a single paper sheet. It can either directly output the modified content to a physical printer or indirectly output it via pdfFactory Pro.
- f) ***pdfFactory Pro Requirements*** (on page 144) - Author-it can "print" content to pdfFactory Pro which outputs the content to a physical printer or saves it to a file.
- g) ***Screen Capture Requirements*** (on page 145) - Author-it can import content (bit-mapped graphic images) from various third-party tools.

"Preview" on Mac OS X - jpg or png format.

"Snagit" on Microsoft Windows - jpg or png format.

"Screenshot" on Linux - jpg or png format.

3 Author-it provides the following services for publishing to:

a) PRINT

Microsoft Word Documents - The Microsoft Word versions contain Author-it specific markups (such as outline numbered sections, page headings, page footers and hyper-links). 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.

PDF Documents - Those Microsoft Word versions converted to PDF format retain Author-it specific markups (such as outline numbered sections, page headings, page footers and hyper-links).

b) WEB

HTML Pages - Feature has not been used.

Java Help - Feature has not been used.

Oracle Help for Java - Feature has not been used.

Website Manager - Feature has not been used.

XHTML Pages - Feature has not been used.

c) HELP

HTML Help - Feature has not been used.

Windows Help - Feature has not been used.

d) XML

DITA - Feature has not been used.

XML - Feature has not been used.

4 Author-it version 5 is out-dated. In 2007 a single user contract was costly to upgrade and cover with its annual maintenance agreement. Recent versions are marketed to enterprise class customers as a cloud service for at least 10 authors.

Version 5 has several flaws:

- a) It incorporates an evaluation version, now expired, of a third-party tool to re-size bit-mapped graphic images. Initially, one could ignore the warning. Since the expiration, one may only use Microsoft Word to adjust the image size.
 - b) Author-it 5.5 was reported to be the last version that would use the Microsoft Jet database engine. Newer versions were reportedly going to use an SQL-type database.
 - c) Table rows which span multiple pages are truncated when printed and do not fully appear in the output file. It is suggested that one split the row and cut and paste the information as appropriate.
 - d) Table layout changes (row insert, delete or move above or below) may or may not work as expected. It is suggested that one cut and paste the row contents from the old to new location.
 - e) Table cell contents vertical alignment changes may or may not work consistently.
 - f) Several section header templates are as yet undefined and produce warning messages in the output file. It is suggested that one use Microsoft Word to substitute the correct information.
- 5** Author-it provides a publish to a standard PDF document. It will use whichever PDF service, such as pdfFactory Pro version 4.70, is available.
- 6** Author-it provides a publish to Microsoft Word document. By designating FinePrint version 7.10 as the output device instead of a physical printer, one can add water mark "Page Tags" such as "Confidential" and "Draft") to the printer input file. FinePrint can also re-organize the printer input file so as to have multiple scaled down pages on each sheet of paper before it transfer the printer input file to pdfFactory Pro.

To capture all bit-mapped graphic images it is simply a matter of:

- a) Opening the Microsoft Word document, after it has been generated by Author-it.
 - b) Select View -> Print Layout
 - c) Select Zoom -> 10%
 - d) Select Printer output to FinePrint
 - e) When FinePrint has completed its somewhat lengthy activities, select Printer output to pdfFactory Pro
 - f) When pdfFactory Pro has completed its somewhat lengthy activities, select output to the disk file and select the file name
 - g) Copy the published Adobe PDF files and Microsoft Office directories to the associated directories in the "tsDocGUI" directory
- 7** Microsoft Word (2002 and 2013 versions) can view and edit the Author-it produced Microsoft Word document.

Due to Author-it's higher level abstraction and support of relocatable shared library documents, Microsoft Word may not properly recognize and handle subsequent Microsoft Word changes to the Author-it numbered sections, paragraphs, lists and hyper-links.

- 8** Microsoft Word can also save the Author-it produced Microsoft Word document as "Plain Text", "MS-DOS Text with Layout" and "Text with Layout".

In an attempt to produce the README and associated release documents that might be examined during a login session to a local or remote computer via an xterm with only a 60-80 column, character-mode display, each of the "save as" options has been tried.

The results invariably required extensive editing to restore the stripped out white space that had separated and indented paragraphs and list entries.

Editing was also needed to fold the white space expanded lines so as to stay within the 60-80 column display margin.

- 9 Apple Computer's Mac OS X offered an office suite ("iWork" '09 which contains "Pages" 4.0, "Numbers" 2.0 and "Keynote" 5.0) which are somewhat compatible with their Microsoft Office counterparts (Word, Excel and Powerpoint).
-

Neither the 2009 or 2013 versions of Pages" can view and edit the Author-it produced Microsoft Word document. Though they can import Microsoft Word documents, they do not properly recognize and handle even the original Author-it markup. The 2013 version does NOT recognize several of the graphic bit-mapped image types.

- 10 A third party offers users of various Linux, Mac OS X and Microsoft Windows distributions, an optional document authoring tool, LibreOffice Version 4.x.

- a) Its Base component can somewhat view the Microsoft Access Database documents.
 - b) Its Calc component can somewhat view and edit the Microsoft Excel Spreadshett documents.
 - c) Its Draw component can somewhat view and edit the Microsoft Viso Drawing documents.
 - d) Its Writer component can can somewhat view and edit the Microsoft Word documents.
-

Though it can import Microsoft Word documents, LibreOffice does not properly recognize and handle even the original Author-it markup.

However, even with its limitations, LibreOffice can be used as a Word Document viewer/reader on non-Microsoft Windows platforms.

4.2.2.3.3 Microsoft Access Requirements

Excerpt From Wikipedia, the free encyclopedia:

"Microsoft Access, also known as Microsoft Office Access, is a database management system from Microsoft that combines the relational Microsoft Jet Database Engine with a graphical user interface and software-development tools. It is a member of the Microsoft Office suite of applications, included in the Professional and higher editions or sold separately. Access stores data in its own format based on the Access Jet Database Engine. It can also import or link directly to data stored in other applications and databases.[1]

Software developers and data architects can use Microsoft Access to develop application software, and "power users" can use it to build software applications. Like other Office applications, Access is supported by Visual Basic for Applications, an object-oriented programming language that can reference a variety of objects including DAO (Data Access Objects), ActiveX Data Objects, and many other ActiveX components. Visual objects used in forms and reports expose their methods and properties in the VBA programming environment, and VBA code modules may declare and call Windows operating-system functions."

4.2.2.3.4 Microsoft Excel Requirements

Excerpt From Wikipedia, the free encyclopedia:

"Microsoft Excel is a spreadsheet application developed by Microsoft for Microsoft Windows and Mac OS. It features calculation, graphing tools, pivot tables, and a macro programming language called Visual Basic for Applications. It has been a very widely applied spreadsheet for these platforms, especially since version 5 in 1993, and it has replaced Lotus 1-2-3 as the industry standard for spreadsheets. Excel forms part of Microsoft Office.

Features

Basic operation

Microsoft Excel has the basic features of all spreadsheets,[2] using a grid of cells arranged in numbered rows and letter-named columns to organize data manipulations like arithmetic operations. It has a battery of supplied functions to answer statistical, engineering and financial needs. In addition, it can display data as line graphs, histograms and charts, and with a very limited three-dimensional graphical display. It allows sectioning of data to view its dependencies on various factors for different perspectives (using pivot tables and the scenario manager).[3] It has a programming aspect, Visual Basic for Applications, allowing the user to employ a wide variety of numerical methods, for example, for solving differential equations of mathematical physics,[4][5] and then reporting the results back to the spreadsheet. It also has a variety of interactive features allowing user interfaces that can completely hide the spreadsheet from the user, so the spreadsheet presents itself as a so-called application, or decision support system (DSS), via a custom-designed user interface, for example, a stock analyzer,[6] or in general, as a design tool that asks the user questions and provides answers and reports.[7][8][9] In a more elaborate realization, an Excel application can automatically poll external databases and measuring instruments using an update schedule,[10] analyze the results, make a Word report or PowerPoint slide show, and e-mail these presentations on a regular basis to a list of participants.

Microsoft allows for a number of optional command-line switches to control the manner in which Excel starts.[11]

4.2.2.3.5 Microsoft Visio Requirements

Excerpt From Wikipedia, the free encyclopedia:

"Microsoft Visio (formerly Microsoft Office Visio) is a diagramming and vector graphics application and is part of the Microsoft Office suite. The product was first introduced in 1992, made by the Shapeware corporation. It was acquired by Microsoft in 2000.

Features

Microsoft Visio 2010 for Windows is available in three editions: Standard, Professional and Premium. The Standard and Professional editions share the same interface, but the latter has additional templates for more advanced diagrams and layouts, as well as unique capabilities intended to make it easy for users to connect their diagrams to data sources and display their data graphically.[2][3] The Premium edition features three additional diagram types as well as intelligent rules, validation, and subprocess (diagram breakdown).[4]

FILE TYPE	NATIVE FILE FORMAT
VSD	Drawing
VSS	Stencil
VST	Template
VSW	Web drawing
VDX	XML drawing (Discontinued[5])
VSX	XML stencil (Discontinued[5])
VTX	XML template (Discontinued[5])
VSDX	OPC/XML drawing[5]
VSDM	OPC/XML drawing, macro-enabled[5]
VSSX	OPC/XML stencil[5]
VSSM	OPC/XML stencil, macro-enabled[5]
VSTX	OPC/XML template[5]
VSTM	OPC/XML template, macro-enabled[5]
VSL	Add-on

Visio 2010 and earlier read and write drawings in VSD or VDX file formats. VSD is the proprietary binary file format used in all of the previous version of Visio. VDX is a well-documented XML Schema-based ("DatadiagramML") format. Visio 2013 drops support for writing VDX files in favor of the new VSDX and VSDM file formats.[5] Created based on Open Packaging Conventions (OPC) standard (ISO 29500, Part 2), a VSDX or VSDM file consists of a group of XML files archived inside a Zip file.[5] The only difference between VSDX and VSDM is that VSDM files may contain macros.[5] Since these files are susceptible to macro virus infection, the program enforces strict security on them.[6]VSD files use LZW-like lossless compression, VDX is not compressed. Hence, a VDX file is typically 3 to 5 times larger.[citation needed] VSDX and VSDM files use the same compression as Zip files. Visio 2010 and earlier use VSD by default. Visio 2013 default is VSDX.

History

Visio began as a standalone product produced by Shapeware Corporation; version 1.0 shipped in 1992. A pre-release, Version 0.92, was distributed free on a floppy disk along with a Microsoft Windows systems readiness evaluation utility. In 1995, Shapeware Corporation changed their name to Visio Corporation to take advantage of market recognition and related product equity.[7] Microsoft acquired Visio in 2000, re-branding it as a Microsoft Office application, like Microsoft Project; however, it has never been included in any of the Office suites. Microsoft included a Visio for Enterprise Architects edition with some editions of Visual Studio .NET 2003 and Visual Studio 2005.[8]with Microsoft Visio 2002 Professional, Microsoft introduced Visio Enterprise Network Tools and Visio Network Center. Visio Enterprise Network Tools was an add-on product that enabled automated network and directory services diagramming. Visio Network Center was a subscription-based website where users could locate the latest network documentation content and exact-replica network equipment shapes from 500 leading manufacturers.[9] The former has been discontinued, while the latter's shape-finding features are now integrated into the program itself.[10] Visio 2007 was released on November 30, 2006.Visio adopted ribbons in its user interface in Visio 2010.[11] Microsoft Word, Excel, PowerPoint, Access and Outlook (to some extents) had already adopted ribbon with the release of Microsoft Office 2007.[12]"

4.2.2.3.6 Microsoft Word Requirements

Excerpt From Wikipedia, the free encyclopedia:

"Microsoft Word is a word processor developed by Microsoft. It was first released in 1983 under the name Multi-Tool Word for Xenix systems.[3][4][5] Subsequent versions were later written for several other platforms including IBM PCs running DOS (1983), Apple Macintosh running Mac OS (1985), AT&T Unix PC (1985), Atari ST (1988), SCO Unix (1994), OS/2 (1989), and Microsoft Windows (1989). Commercial versions of Word are licensed as a standalone product or as a component of Microsoft Office, Windows RT or the discontinued Microsoft Works suite. Freeware editions of Word are Microsoft Word Viewer and Office Online, both of which have limited features.

History[edit]

Origins and growth[edit]

In 1981, Microsoft hired Charles Simonyi, the primary developer of Bravo, the first GUI word processor, which was developed at Xerox PARC.[6] Simonyi started work on a word processor called Multi-Tool Word and soon hired Richard Brodie, a former Xerox intern, who became the primary software engineer.[6][7][8]announced Multi-Tool Word for Xenix[6] and MS-DOS in 1983.[9] Its name was soon simplified to Microsoft Word.[3] Free demonstration copies of the application were bundled with the November 1983 issue of PC World, making it the first to be distributed on-disk with a magazine.[3][10] That year Microsoft demonstrated Word running on Windows.[11]

Unlike most MS-DOS programs at the time, Microsoft Word was designed to be used with a mouse.[9] Advertisements depicted the Microsoft Mouse, and described Word as a WYSIWYG, windowed word processor with the ability to Undo and display bold, italic, and underlined text,[12] although it could not render fonts.[3] It was not initially popular, since its user interface was different from the leading word processor at the time, WordStar.[13] However, Microsoft steadily improved the product, releasing versions 2.0 through 5.0 over the next six years. In 1985, Microsoft ported Word to Mac OS. This was made easier by Word for DOS having been designed for use with high-resolution displays and laser printers, even though none were yet available to the general public.[14] Following the precedents of LisaWrite and MacWrite, Word for Mac OS added true WYSIWYG features. After its release, Word for Mac OS's sales were higher than its MS-DOS counterpart for at least four years.[6]

The second release of Word for Mac OS, shipped in 1987, was named Word 3.0 to synchronize its version number with Word for DOS; this was Microsoft's first attempt to synchronize version numbers across platforms. Word 3.0 included numerous internal enhancements and new features, including the first implementation of the Rich Text Format (RTF) specification, but was plagued with bugs. Within a few months, Word 3.0 was superseded by a more stable Word 3.01, which was mailed free to all registered users of 3.0.[14] After MacWrite, Word for Mac OS never had any serious rivals. Word 5.1 for Mac OS, released in 1992, was a very popular word processor owing to its elegance, relative ease of use and feature set. Many users say it is the best version of Word for Mac OS ever created.[14][15]

In 1986, an agreement between Atari and Microsoft brought Word to the Atari ST[16] under the name Microsoft Write. The Atari ST version was a port of Word 1.05 for the Mac OS[17][18] and was never updated.

The first version of Word for Windows was released in 1989. With the release of Windows 3.0 the following year, sales began to pick up and Microsoft soon became the market leader for word processors for IBM PC-compatible computers.[6] In 1991, Microsoft capitalized on Word for Windows' increasing popularity by releasing a version of Word for DOS, version 5.5, that replaced its unique user interface with an interface similar to a Windows application.[19][20] When Microsoft became aware of the Year 2000 problem, it made Microsoft Word 5.5 for DOS available for download free. As of March 2014, it is still available for download from Microsoft's web site.[21] In 1991, Microsoft embarked on a project code-named Pyramid to completely rewrite Microsoft Word from the ground up. Both the Windows and Mac OS versions would start from the same code base. It was abandoned when it was determined that it would take the development team too long to rewrite and then catch up with all the new capabilities that could have been added in the same time without a rewrite. Instead, the next versions of Word for Windows and Mac OS, dubbed version 6.0, both started from the code base of Word for Windows 2.0.[15]

With the release of Word 6.0 in 1993, Microsoft again attempted to synchronize the version numbers and coordinate product naming across platforms, this time across DOS, Mac OS, and Windows (this was the last version of Word for DOS). It introduced AutoCorrect, which automatically fixed certain typing errors, and AutoFormat, which could reformat many parts of a document at once. While the Windows version received favorable reviews (e.g.,[22]), the Mac OS version was widely derided. Many accused it of being slow, clumsy and memory intensive, and its user interface differed significantly from Word 5.1.[15] In response to user requests, Microsoft offered Word 5 again, after it had been discontinued.[23] Subsequent versions of Word for Mac OS X are no longer direct ports of Word for Windows, instead featuring a mixture of ported code and native code.

Word for Windows[edit]

A full-featured word processing program for Windows and Mac OS X from Microsoft. Available stand-alone or as part of the Microsoft Office suite, Word contains rudimentary desktop publishing capabilities and is the most widely used word processing program on the market. Word files are commonly used as the format for sending text documents via e-mail because almost every user with a computer can read a Word document by using the Word application, a Word viewer or a word processor that imports the Word format (see Microsoft Word Viewer). Word 95 for Windows was the first 32-bit version of the product, released with Office 95 around the same time as Windows 95. It was a straightforward port of Word 6.0 and it introduced few new features, one of them being red-squiggle underlined spell-checking.[24] Starting with Word 95, releases of Word were named after the year of its release, instead of its version number.[25]

Word for Mac[edit]

In 1997, Microsoft formed the Macintosh Business Unit as an independent group within Microsoft focused on writing software for Mac OS. Its first version of Word, Word 98, was released with Office 98 Macintosh Edition. Document compatibility reached parity with Word 97,[23] and it included features from Word 97 for Windows, including spell and grammar checking with squiggles.[26] Users could choose the menus and keyboard shortcuts to be similar to either Word 97 for Windows or Word 5 for Mac OS.

Word 2001, released in 2000, added a few new features, including the Office Clipboard, which allowed users to copy and paste multiple items.[27] It was the last version to run on classic Mac OS and, on Mac OS X, it could only run within the Classic Environment. Word X, released in 2001, was the first version to run natively on, and required, Mac OS X,[26] and introduced non-contiguous text selection.[28]

Word 2004 was released in May 2004. It included a new Notebook Layout view for taking notes either by typing or by voice.[29] Other features, such as tracking changes, were made more similar with Office for Windows.[30]

Word 2008, released on January 15, 2008, included a Ribbon-like feature, called the Elements Gallery, that can be used to select page layouts and insert custom diagrams and images. It also included a new view focused on publishing layout, integrated bibliography management,[31] and native support for the new Office Open XML format. It was the first version to run natively on Intel-based Macs.[32]

Word 2010 allows more customization of the Ribbon,[33] adds a Backstage view for file management,[34] has improved document navigation, allows creation and embedding of screenshots,[35] and integrates with Word Web App.[36]

Word 2011, released in October 2010, replaced the Elements Gallery in favor of a Ribbon user interface that is much more similar to Office for Windows,[37] and includes a full-screen mode that allows users to focus on reading and writing documents, and support for Office Web Apps.[38]"

4.2.2.3.7 FinePrint Requirements

Excerpt from FinePrint Web Page:

"Provides a wealth of print options you may have been wishing for - and some you didn't even know you wanted - while taking the mystery out of complex print jobs." - PC MAGAZINE

Features

- Universal print previewer.
- Delete unwanted pages
- Convert to grayscale
- Lighten content to save ink
- Remove blank pages
- Crop pages
- Edit text
- Remove unwanted text and images
- Print multiple pages on a single sheet
- Print electronic letterhead

4.2.2.3.8 pdfFactory Pro Requirements

Excerpt from FinePrint Web Page:

"When it comes to sheer simplicity, flexibility, and accuracy, nothing compares to pdfFactory Pro. Rating: 5 stars" – PC PRO, RECOMMENDED

Features

- Create PDFs on letterhead.
- Add page numbering, headers, footers, watermarks.
- Bookmark each job automatically.
- Create table of contents automatically.
- Convert to grayscale.
- PDF/A archiving.
- Add text notes.
- Fill in forms.
- Crop pages.
- Edit text.
- Add multiple signatures and initials with Notes function
- Combine documents into a single PDF and rearrange them.
- Custom Drivers for specific tasks.
- Text highlight, copy and redact.
- Graphic copy, delete, save.
- Convert text to links.

4.2.2.3.9 Screen Capture Requirements

From Wikipedia, the free encyclopedia:

"A screenshot, screen capture (or screen-cap), screen dump, screengrab[1] is an image taken by the computer user to record the visible items displayed on the monitor, television, or another visual output device. Usually, this is a digital image using the operating system or software running on the computer, but it can also be a capture made by a camera[2] or a device intercepting the video output of the display.

In the 1980s, computer operating systems did not universally have built-in functionality for capturing screenshots. Some text-only screens could be dumped to a text file, but the result would only capture the content of the screen, not the appearance, nor were graphics screens preservable this way. Some systems had a BSAVE command to capture the area of memory where screen data was stored, but this required access to a BASIC prompt. Screenshot kits were available for standard (film) cameras that included a long hood to attach between the screen and camera lens, as well as a closeup lens for the camera. Polaroid film was popular for capturing screenshots, because of the instant results and close-focusing capability of Polaroid cameras.

Screenshots can be used to demonstrate a program, a particular problem a user might be having, or generally when display output needs to be shown to others or archived. For example, after being emailed a screenshot, a Web page author might be surprised to see how their page looks on a different Web browser and can take corrective action. Likewise with differing email software programs, (particularly such as in a cell phone, tablet, etc.,) a sender might have no idea how their email looks to others until they see a screenshot from another computer and can (hopefully) tweak their settings appropriately."

1 Linux

On KDE or GNOME, PrtScr key behavior is quite similar to Windows. (See § Microsoft Windows.) In addition, the following screenshooting utilities are bundled with Linux distributions:

- a) GIMP: A raster graphics editor that can take screenshots too[8]
- b) ImageMagick: Has an "import" command-line tool that captures screenshots in a variety of formats. Type `import -window root ~/screenshot.png` to capture the entire screen to your home directory.
- c) KSnapshot: The default screen grabbing utility in the KDE
- d) `gnome-screenshot`: The default screen grabbing utility in GNOME
- e) `xwd`: The screen capture utility of the X Window System

2 Mac OS X

On Mac OS X, a user can take a screenshot of an entire screen by pressing $\text{⌘} + \text{⇧} + \text{⌃}$, or of a chosen area of the screen by $\text{⌘} + \text{⇧} + \text{⌃} + \text{⌘}$. This screenshot is saved to the user's desktop, with one PNG file per attached monitor. If the user holds down `Ctrl` while doing either then the screenshot will be copied to the clipboard instead.

Beginning with Mac OS X Panther, it is possible to make a screenshot of an active application window. By following $\text{⌘} + \text{⇧} + \text{⌃} + \text{⌘}$, with pressing the Spacebar, the cross-hair cursor turns into a small camera icon. The current window under the cursor is highlighted, and a click on the mouse or trackpad will capture a screenshot of the entire highlighted element (including the parts offscreen or covered by other windows).[9]

A provided application called Grab will capture a chosen area, a whole window, the whole screen, or the whole screen after 10 seconds and pops the screenshot up in a window ready for copying to the clipboard or saving as a TIFF. The Preview application, also provided, has the same capture options as Grab but opens the captured image immediately in a new window.

A shell utility called "screencapture" (located in /usr/sbin/screencapture) can be used from the Terminal application or in shell scripts to capture screenshots and save them to files. Various options are available to choose the file format of the screenshot, how the screenshot is captured, if sounds are played, etc. This utility might only be available when the Mac OS X developer tools are installed. A user cannot capture the screen while DVD Player is running.

3 Windows

On Windows, pressing PrtScr captures a screenshot of the entire desktop, while Alt+PrtScr captures only the active window. Captured screenshots do not include the mouse pointer. Windows places these captured screenshots in the clipboard, meaning that an additional program needs to retrieve them from the clipboard. Starting with Windows 8, however, \boxplus Win+PrtScr or \boxplus Win+Volume up instantly saves a screenshot to the "Screenshots" folder in "Pictures" library. All screenshots are saved as PNG files.[10]

Windows Vista and later include a utility called Snipping Tool, first introduced in Windows XP Tablet PC Edition. It is a screen-capture tool that allows taking screenshots ("snips") of a window, rectangular area, or free-form area. Snips can then be annotated, saved as an image file or as an HTML page, or emailed. However, it does not work with non-tablet XP versions but represents an XP compatible equivalent. Windows 7 and later also include Problem Step Recorder as part of their troubleshooting platforms that once started, automatically captures a screenshot at mouse clicks.[11]

There are exceptions to what can be captured by this method. For example, contents in hardware overlay are not captured. This includes video images that Windows Media Player 10 or earlier play. As such, special software may be required to capture the screens of video games.[12]

4 External tools

- a) IrfanView
- b) Microsoft Word
- c) Snagit

4.2.2.4 Document Publication Standards

The free and open source publication of the "tsWxGTUI_PyVx" Toolkit software products shall be subject to the following terms and conditions:

1 Software Source Code

GNU General Public License (GPL), Version 3, 29 June 2007

- a) Python 2.x programming language syntax formats
- b) Python 3.x programming language syntax formats

2 Software User Documents

GNU Free Documentation License (GFDL) 1.3, 3 November 2008

- a) Engineering Documents in Adobe PDF and Microsoft Office Formats

- b) Operator Documents in Plain Text Formats

4.2.3 Reusable Software Products

This paragraph shall be divided into the following subparagraphs.

- 1** *Incorporating Reusable Software Products* (on page 147)
- 2** *Developing Reusable Software Products* (on page 148)

4.2.3.1 Incorporating Reusable Software Products

This paragraph shall describe the approach to be followed for identifying, evaluating, and incorporating reusable software products, including the scope of the search for such products and the criteria to be used for their evaluation. It shall cover all contractual clauses concerning this topic. Candidate or selected reusable software products known at the time this plan is prepared or updated shall be identified and described, together with benefits, drawbacks, and restrictions, as applicable, associated with their use.

1 Identification Approach

"Identifying and Qualifying Reusable Software Components" by Gianluigi Caldiera and Victor R. Basili, University of Maryland, College Park, Maryland, February 1991.

See <http://www.cs.umd.edu/~basili/publications/journals/J41.pdf>

- a) Popular means to provide required functional and interface capabilities.
- b) Available as free and open software under OSI Approved License.
- c) Track record of accomplishments in development, maintenance and support over an extended period of time.
- d) Suitable for cross-platform deployment.

2 Evaluation Approach

- a) Download candidates from trusted source
- b) Install candidates on development and test platforms
- c) Operate candidates regularly over several months
- d) Choose the candidate that best fulfills the "tsWxGTUI_PyVx" Toolkit project requirements.

3 Incorporation Approach

- a) Include the selected candidate in the "tsWxGTUI_PyVx" Toolkit distribution release.

4.2.3.2 Developing Reusable Software Products

This paragraph shall describe the approach to be followed for identifying, evaluating, and reporting opportunities for developing reusable software products. It shall cover all contractual clauses concerning this topic.

TBD - This section under construction.

Draft

4.2.4 Handling Of Critical Requirements

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for handling requirements designated critical. The planning in each subparagraph shall cover all contractual clauses concerning the identified topic.

TBD - This section under construction.

```

+-----+-----+ TeamSTARS "tsWxGTUI_PyVx" Toolkit
| ts | Wx | with Python-based
+-----+-----+ Command Line Interface (CLI)
| G T U I | and "wxPython"-style, "nCurses"-based
+-----+-----+ Graphical-Text User Interface (GUI)

Get that cross-platform, pixel-mode "wxPython" feeling
on character-mode terminals and terminal emulators.
-----

                                PREFACE

The supplier of a turn-key system used by an end-user
is the "system integrator" and therefore assumes all
responsibility for any express or implied warranties,
including, but not limited to, the implied warranties of
merchantability and fitness for a particular purpose.

Only the "system integrator" can have the facilities to
iteratively engineer, test, qualify and certify the
turn-key system before it is used by the end-user.

The building block developers of the TeamSTARS
"tsWxGTUI_PyVx" Toolkit and any of its Commercially
available Off-The-Shelf (COTS) components cannot iter-
atively engineer, test, qualify and certify the turn-
key system because they neither know its requirements
nor do they have the facilities to qualify and certify
all conceivable system configurations that might
incorporate their components.
-----

                                DISCLAIMER

THIS SOFTWARE IS NOT FAULT TOLERANT AND SHOULD NOT BE
USED IN ANY SITUATION ENDANGERING HUMAN LIFE OR PROP-
ERTY.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND

```

CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANT-
TIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANT-
TIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
HOLDERS AND CONTRIBUTORS BE LIABLE FOR ANY DIRECT, IN-
DIRECT, 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.

4.2.5 Computer Hardware Resource Utilization

This paragraph shall describe the approach to be followed for allocating computer hardware resources and monitoring their utilization. It shall cover all contractual clauses concerning this topic.

TBD - This section under construction.

4.2.6 Recording Rationale

This paragraph shall describe the approach to be followed for recording rationale that will be useful to the support agency for key decisions made on the project. It shall interpret the term "key decisions" for the project and state where the rationale are to be recorded. It shall cover all contractual clauses concerning this topic.

TBD - This section under construction.

4.2.7 Access For Acquirer Review

This paragraph shall describe the approach to be followed for providing the acquirer or its authorized representative access to developer and subcontractor facilities for review of software products and activities. It shall cover all contractual clauses concerning this topic.

TBD - This section under construction.

5 PLANS FOR PERFORMING DETAILED SOFTWARE DEVELOPMENT ACTIVITIES

This section shall be divided into the following paragraphs. Provisions corresponding to non-required activities may be satisfied by the words "Not applicable." If different builds or different software on the project require different planning, these differences shall be noted in the paragraphs. The discussion of each activity shall include the approach (methods/procedures/tools) to be applied to: 1) the analysis or other technical tasks involved, 2) the recording of results, and 3) the preparation of associated deliverables, if applicable. The discussion shall also identify applicable risks/uncertainties and plans for dealing with them. Reference may be made to 4.2.1 if applicable methods are described there.

- 1 *Project Planning And Oversight* (on page 152)
- 2 *Establishing A Software Development Environment* (on page 153)
- 3 *System Requirements Analysis* (on page 154)
- 4 *Software Design* (on page 154)
- 5 *Software Implementation And Unit Testing* (on page 155)
- 6 *Unit Integration And Testing* (on page 155)
- 7 *CSCI Qualification Testing* (on page 156)
- 8 *CSCI/HWCI Integration And Testing* (on page 156)
- 9 *System Qualification Testing* (on page 157)
- 10 *Preparing For Software Use* (on page 158)
- 11 *Preparing For Software Transition* (on page 159)
- 12 *Software Configuration Management* (on page 160)
- 13 *Software Product Evaluation* (on page 160)
- 14 *Software Quality Assurance* (on page 161)
- 15 *Corrective Action* (on page 161)
- 16 *Joint Technical and Management Reviews* (on page 162)
- 17 *Other Software Development Activities* (on page 162)

5.1 Project Planning And Oversight

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for project planning and oversight. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

- 5.1.1 Software development planning (covering updates to this plan)
 - 5.1.2 CSCI test planning
 - 5.1.3 System test planning
 - 5.1.4 Software installation planning
 - 5.1.5 Software transition planning
 - 5.1.6 Following and updating plans, including the intervals for management review
-

5.1.1 Software Development Planning

5.1.2 CSCI Test Planning

5.1.3 System Test Planning

5.1.4 Software Installation Planning

5.1.5 Software Transition Planning

5.1.6 Following and Updating Plans

5.2 Establishing A Software Development Environment

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for establishing, controlling, and maintaining a software development environment. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

- 5.2.1 Software engineering environment
 - 5.2.2 Software test environment
 - 5.2.3 Software development library
 - 5.2.4 Software development files
 - 5.2.5 Non-deliverable software
-

5.3 System Requirements Analysis

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for participating in system requirements analysis. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

5.3.1 Analysis of user input

5.3.2 Operational concept

5.3.3 System requirements

5.4 Software Design

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for software design. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

5.6.1 CSCI-wide design decisions

5.6.2 CSCI architectural design

5.6.3 CSCI detailed design

5.5 Software Implementation And Unit Testing

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for software implementation and unit testing. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

- 5.7.1 Software implementation
 - 5.7.2 Preparing for unit testing
 - 5.7.3 Performing unit testing
 - 5.7.4 Revision and retesting
 - 5.7.5 Analyzing and recording unit test results
-

5.6 Unit Integration And Testing

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for unit integration and testing. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

- 5.8.1 Preparing for unit integration and testing
 - 5.8.2 Performing unit integration and testing
 - 5.8.3 Revision and retesting
 - 5.8.4 Analyzing and recording unit integration and test results
-

5.7 CSCI Qualification Testing

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for CSCI qualification testing. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

- 5.9.1 Independence in CSCI qualification testing
 - 5.9.2 Testing on the target computer system
 - 5.9.3 Preparing for CSCI qualification testing
 - 5.9.4 Dry run of CSCI qualification testing
 - 5.9.5 Performing CSCI qualification testing
 - 5.9.6 Revision and retesting
 - 5.9.7 Analyzing and recording CSCI qualification test results
-

5.8 CSCI/HWCI Integration And Testing

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for participating in CSCI/HWCI integration and testing. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

- 5.10.1 Preparing for CSCI/HWCI integration and testing
 - 5.10.2 Performing CSCI/HWCI integration and testing
 - 5.10.3 Revision and retesting
 - 5.10.4 Analyzing and recording CSCI/HWCI integration and test results
-

5.9 System Qualification Testing

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for participating in system qualification testing. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

- 5.11.1 Independence in system qualification testing
 - 5.11.2 Testing on the target computer system
 - 5.11.3 Preparing for system qualification testing
 - 5.11.4 Dry run of system qualification testing
 - 5.11.5 Performing system qualification testing
 - 5.11.6 Revision and retesting
 - 5.11.7 Analyzing and recording system qualification test results
-

5.10 Preparing For Software Use

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for preparing for software transition. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

5.13.1 Preparing the executable software

5.13.2 Preparing source files

5.13.3 Preparing version descriptions for the support site

5.13.4 Preparing the "as built" CSCI design and other software support information

5.13.5 Updating the system design description

5.13.6 Preparing support manuals

5.13.7 Transition to the designated support site

5.11 Preparing For Software Transition

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for preparing for software transition. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

5.13.1 Preparing the executable software

5.13.2 Preparing source files

5.13.3 Preparing version descriptions for the support site

5.13.4 Preparing the "as built" CSCI design and other software support information

5.13.5 Updating the system design description

5.13.6 Preparing support manuals

5.13.7 Transition to the designated support site

5.12 Software Configuration Management

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for software configuration management. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

- 5.14.1 Configuration identification
 - 5.14.2 Configuration control
 - 5.14.3 Configuration status accounting
 - 5.14.4 Configuration audits
 - 5.14.5 Packaging, storage, handling, and delivery
-

5.13 Software Product Evaluation

This paragraph shall be divided into the following sub\paragraphs to describe the approach to be followed for software product evaluation. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

- 5.15.1 In-process and final software product evaluations
 - 5.15.2 Software product evaluation records, including items to be recorded
 - 5.15.3 Independence in software product evaluation
-

5.14 Software Quality Assurance

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for software quality assurance. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

5.16.1 Software quality assurance evaluations

5.16.2 Software quality assurance records, including items to be recorded

5.16.3 Independence in software quality assurance

5.15 Corrective Action

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for corrective action. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

5.17.1 Problem/change reports, including items to be recorded (candidate items include project name, originator, problem number, problem name, software element or document affected, origination date, category and priority, description, analyst assigned to the problem, date assigned, date completed, analysis time, recommended solution, impacts, problem status, approval of solution, follow-up actions, corrector, correction date, version where corrected, correction time, description of solution implemented)

5.17.2 Corrective action system

5.16 Joint Technical and Management Reviews

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for joint technical and management reviews. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

5.18.1 Joint technical reviews, including a proposed set of reviews

5.18.2 Joint management reviews, including a proposed set of reviews

5.17 Other Software Development Activities

This paragraph shall be divided into the following subparagraphs to describe the approach to be followed for other software development activities. The planning in each subparagraph shall cover all contractual clauses regarding the identified topic.

5.19.1 Risk management, including known risks and corresponding strategies

5.19.2 Software management indicators, including indicators to be used

5.19.3 Security and privacy

5.19.4 Subcontractor management

5.19.5 Interface with software independent verification and validation (IV&V) agents

5.19.6 Coordination with associate developers

5.19.7 Improvement of project processes

5.19.8 Other activities not covered elsewhere in the plan

6 SCHEDULES AND ACTIVITY NETWORK

This section shall present the following:

- 1 ***Schedule(s)*** (on page 163) - Identify the activities in each build and showing initiation of each activity, availability of draft and final deliverables and other milestones, and completion of each activity
- 2 ***Activity Network*** (on page 163) - Depiction of sequential relationships and dependencies among activities and identifying those activities that impose the greatest time restrictions on the project.

6.1 Schedule(s)

This section shall present schedule(s) identifying the activities in each build and showing initiation of each activity, availability of draft and final deliverables and other milestones, and completion of each activity.

The *TeamSTARS* "tsWxGTUI_PyVx" Toolkit Pre-alpha Release 0.0.0

- 1 Run "chmod -Rf 755" from within the following directories:
 - a) ./SoftwareGadgetry-PyPI/Python 2.x/
 - b) ./SoftwareGadgetry-PyPI/Python 3.x/
- 2 Run "python setup.py sdist" from within the following directories:
 - a) ./SoftwareGadgetry-PyPI/Python 2.x/
 - b) ./SoftwareGadgetry-PyPI/Python 3.x/

6.2 Activity Network

Draft

7 PROJECT ORGANIZATION AND RESOURCES

This section shall be divided into the following paragraphs to describe the project organization and resources to be applied in each build.

1 *Project Organization* (on page 165)

2 *Project Resources* (on page 166)

7.1 Project Organization

This paragraph shall describe the organizational structure to be used on the project, including the organizations involved, their relationships to one another, and the authority and responsibility of each organization for carrying out required activities.

1 **"tsWxGTUI_PyVx" Toolkit Development Organization(s)**

a) *TeamSTARS*

Provider of feasibility prototype software

b) *Software Gadgetry*

Provider of free, open source prototype software

2 **Commercial Off-The-Shelf (COTS) Product Development Organization(s)**

Independent, third-party providers of computer hardware, software and service products.

a) *Apple*

Provider of the Mac OS X Unix-style operating system for various 32-/64-bit host computer processors.

b) *Darwin*

Provider of the free, open source Linux distributions for various 32-/64-bit host computer processors.

c) *Debian*

Provider of the free, open source Linux distributions for various 32-/64-bit host computer processors.

d) *Free Software Foundation (www.fsf.org)*

Provider of the free, open source ncurses programming library

- e) FreeBSD Foundation
Provider of the free, open source BSD-style Unix (FreeBSD, PC-BSD) distributions for various 32-/64-bit host computer processors.
- f) OpenIndiana
Provider of the free, open source Solaris-style Unix distributions for various 32-/64-bit host computer processors.
- g) Parallels Software
Provider of the hypervisor software that enable the Apple Computer hardware and Mac OS X operating systems to create and concurrently run one or more guest operating systems (such as various Linux, Microsoft Windows and Unix).
- h) Python Software Foundation (www.python.org)
Provider of the free, open source Python virtual machine and programming library for various 32-/64-bit host computer processors and operating systems.
- i) Red Hat
Provider of the free, open source Cygwin (Linux-style Command Line Interface and GNU Toolit add-on for Microsoft Windows) and various Linux (CentOS, Enterprise, Fedora, Red hat, Scientific) distributions.
- j) Ubuntu
Provider of the free, open source Linux distributions for various 32-/64-bit host computer processors.
- k) Wikimedia Foundation (www.wikimedia.org)
Provider of a multilingual, web-based, free-content encyclopedia project supported by the Wikimedia Foundation and based on an openly editable model.
- l) wxWidgets Team (www.wxwidgets.org)
Provider of free, open source, cross-platorm Graphical User Interface Toolkit distributions for various 32-/64-bit host computer processors and operating systems.

7.2 Project Resources

This paragraph shall describe the resources to be applied to the project. It shall include, as applicable:

- 1** *Personnel Resources* (on page 167)
- 2** *Overview of Developer Facilities* (on page 167)
- 3** *Acquirer-furnished Equipment, Software, Services, Documentation, Data, and Facilities* (on page 167)
- 4** *Other Required Resources* (on page 168)

7.2.1 Personnel Resources

Personnel resources, include:

- 1) The estimated staff-loading for the project (number of personnel over time)
 - 2) The breakdown of the staff-loading numbers by responsibility (for example, management, software engineering, software testing, software configuration management, software product evaluation, software quality assurance)
 - 3) A breakdown of the skill levels, geographic locations, and security clearances of personnel performing each responsibility
-

- 1** Estimated Staff-loading for the Project (number of personnel over time)
- 2** Breakdown of the Staff-loading Numbers by Responsibility
 - a) Management
 - b) Software Engineering
 - c) Software Testing
 - d) Software Configuration Management
 - e) Software Product Evaluation
 - f) Software Quality Assurance

7.2.2 Overview of Developer Facilities

Overview of developer facilities to be used, including geographic locations in which the work will be performed, facilities to be used, and secure areas and other features of the facilities as applicable to the contracted effort.

- 1** Geographic Locations in which the Work Will Be Performed
- 2** Facilities To Be Used
- 3** Secure Areas and Other Features Of The Facilities As Applicable To The Contracted Effort.

7.2.3 Acquirer-furnished Equipment, Software, Services, Documentation, Data, and Facilities

Acquirer-furnished equipment, software, services, documentation, data, and facilities required for the contracted effort. A schedule detailing when these items will be needed shall also be included.

- 1** Equipment
- 2** Software
- 3** Services

4 Documentation

5 Data

6 Facilities Required for the Contracted Effort

7.2.4 Other Required Resources

Other required resources, including a plan for obtaining the resources, dates needed, and availability of each resource item.

1 Required Resources, including a plan for obtaining the resources, dates needed, and availability of each resource item

Draft

8 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 any terms and definitions needed to understand this document.

1 TeamSTARS - Feasibility Prototype

started July 2007

transitioned December 2009

a) *Richard S. Gordon*

tsLibCLI, tsToolsCLI, tsLibGUI, tsUtilities

tsDocCLI, tsDocGUI

b) *Frederick A. Kier*

tsLibCLI

2 Software Gadgetry - Free, Open Source Pre-alpha Release 0.0.0

started January 2010

targeted for December 2014

a) *Richard S. Gordon*

tsLibCLI, tsLibGUI, tsToolsCLI, tsUtilities

tsDocCLI, tsDocGUI

8.1 Toolkit Evolution Plans & Design Considerations

This section describes *TeamSTARS* "tsWxGTUI_Py_Vx" Toolkit Evolution Plans & Design Considerations.

***** DRAFT 16-Jan-2016 *****

1 *Terminal Control Library Considerations* (on page 170)

2 *Operator Terminal Design Considerations* (on page 172)

- 3 *Computer Processor Design Considerations* (on page 172)
- 4 *Python Interpreter and Virtual Machine Considerations* (on page 173)
 - a) *Cython Programming Language* (on page 173)
 - b) *Python 2.7.11* (on page 173)
 - c) *Python 3.6.0, Alpha 1* (on page 174)
 - d) *Third-Party Alternatives to "curses"/"ncurses"* (on page 174)
- 5 *Python Application Program Launch Considerations* (on page 174)
- 6 *TeamSTARS "tsWxGTUI_PyVx" Toolkit Considerations* (on page 175)
- 7 *Reference Documentation* (on page 175)

8.1.1 Terminal Control Library Considerations

From Wikipedia, the free encyclopedia:

"History

The first curses library was written by Ken Arnold and originally released with BSD UNIX, where it was used for several games, most notably Rogue. Some improvements were made to the BSD library in the 1990s as "4.4BSD" curses, e.g., to provide more than one type of video highlighting.[citation needed] However, those are not widely used.

The name "curses" is a pun on cursor optimization. Sometimes it is incorrectly stated that curses was used by the vi editor. In fact the code in curses that optimizes moving the cursor from one place on the screen to another was borrowed from vi, which predated curses.

According to Goodheart, Kenneth Arnold's original implementation of curses started by reusing functions from the termcap library, and adding to that. A few years later, Mark Horton, who had made improvements to the vi and termcap sources at Berkeley, went to AT&T Corporation and made a different version using terminfo, which became part of UNIX System III and UNIX System V. Due to licensing restrictions on the latter, the BSD and AT&T versions of the library were developed independently. In addition to the termcap/terminfo improvement, other improvements were made in the AT&T version:

video highlighting (bold, underline)

The BSD version supported only standout.

line-drawing

The BSD version gave little support here.

colors

This was not anticipated in the BSD version.

AT&T curses development appears to have halted in the mid-1990s when X/Open Curses was defined. However, development of ncurses and PDCurses continues. A version of BSD curses continues to be maintained in the NetBSD operating system (wide character support, termcap to terminfo migration, etc.).

pcurses and PDCurses

Main article: PDCurses

Different lines of development started by imitating the AT&T curses, from at least three implementations: pcurses by Pavel Curtis (started in 1982), PDCurses (Public Domain curses) by Mark Hessling to support his editor THE (started in 1987) as well as Rext/Curses, and PC curses (version 1.4 and earlier by Bjorn Larsson based inspired by Pavel Curtis' library before 1990.)

ncurses

Main article: ncurses

ncurses (new curses) *"originated as pcurses ... and was re-issued as ncurses 1.8.1 in late 1993". ncurses is the most widely known implementation of curses, and has motivated further development of other variations, such as BSD curses in the NetBSD project.*

Portability

Although the ncurses library was initially developed under Linux, OpenBSD, FreeBSD, and NetBSD it has been ported to many other ANSI/POSIX UNIX systems, mainly by Thomas Dickey. PDCurses, while not identical to ncurses, uses the same function calls and operates the same way as ncurses does except that PDCurses targets different devices, e.g., console windows for DOS, Win32, OS/2, as well as X11. Porting between the two is not difficult. For example, the roguelike game ADOM was written for Linux and ncurses, later ported to DOS and PDCurses."

For more information see Third-Party Alternatives to "curses"/"ncurses" (on page 174)

Screenshots

Replaced the Wikipedia cited screenshots with TeamSTARS "tsWxGTUI_PyVx" Toolkit specific screenshots.

For more information see SAMPLE SCREEN SHOTS (on page 183)

Curses-based software

Curses-based software is software whose user interface is implemented through the Curses library, or a compatible library (such as Ncurses).

Curses is designed to facilitate GUI-like functionality on a text-only device, such as a PC running in console mode, a hardware ANSI terminal, a Telnet or SSH client, or similar.

Curses-based programs often have a user interface that resembles a traditional graphical user interface, including 'widgets' such as text boxes and scrollable lists, rather than the command line interface (CLI) most commonly found on text-only devices. This can make them more user-friendly than a CLI-based program, while still being able to run on text-only devices. Curses-based software can also have a lighter resource footprint and operate on a wider range of systems (both in terms of hardware and software) than their GUI-based counterparts. This includes old pre-1990 machines along with modern embedded systems using text-only displays.

However, not all Curses-based software employs a text user interface which resembles a graphical user interface. One counterexample would be the popular vi text editor, which while not being CLI-based, uses memorized keyboard commands almost exclusively, rather than the prompting TUI/GUI style, which relies more on recognition than recall.

Curses is most commonly associated with Unix-like operating systems, although implementations for Microsoft Windows also exist."

8.1.2 Operator Terminal Design Considerations

The TeamSTARS "tsWxGTUI_PyVx" Toolkit was implemented in both the mature Python 2.x and evolving Python 3.x programming languages which include a curses-style text-mode terminal control library module for such POSIX-compliant platforms as GNU/Linux, Mac OS X and Unix.

It only supports those Microsoft Windows platforms which installed Cygwin, the free GNU/Linux-like command line interface and GNU toolchain plug-in made by Red Hat.

On platforms with 32-bit processors, the Python 2x and Python 3x curses library modules are designed to interface with the host platform's curses or ncurses library:

1 **BSD UNIX "curses"** (proprietary)

Gets terminal capability from the "termcap" software library and database

2 **AT&T UNIX System V "curses"** (proprietary)

Gets terminal capabilities and more from the "terminfo" (formerly "termcap") database. For almost every model of terminal it tells application programs what the terminal is capable of doing.

3 **Free Software Foundation "ncurses"** 5.0-5.9 and 6.0-6.x (non-proprietary)

Gets terminal capabilities and more from "terminfo" database

8.1.3 Computer Processor Design Considerations

1 The 32-bit processor data word size is large enough to support:

- a) wheel mouse (mouse #5)
- b) displays with non-color (vt100) or at least 8 (xterm) or no more than 16-colors (xterm-16color with upto 256-color pairs).

2 The 64-bit processor data word size is large enough to support:

- a) wheel mouse (mouse #5)
- b) displays with non-color (vt100) or at least 8 (xterm) or more than 256-colors (xterm-256color, or higher, with more than 65,536-color pairs).

8.1.4 Python Interpreter and Virtual Machine Considerations

Python 2x and Python 3x provide a `curses` library module. It is designed to interface with the host platform's "`curses`"/"`ncurses`" terminal control library.

Preserving the use of the Python `curses` library module might maintain upgradeability to the 8-Aug-2015 release of `ncurses` 6.0. It would also avoid the need for users to install and configure Third Party alternatives to "`curses`"/"`ncurses`" which would then require changes to toolkit & application programming and the inevitable troubleshooting.

The Python `curses` library module is implemented in Cython. It provides a usable subset of the "`curses`"/"`ncurses`" Application Programming Interface (API).

8.1.4.1 Cython Programming Language

The Cython programming language is a superset of Python with a foreign function interface for invoking C/C++ routines and the ability to declare the static type of subroutine parameters and results, local variables, and class attributes.

It would require considerable research and expertise to design, implement and debug the extensive changes to the Python `curses` library module needed to unofficially support the complete "`curses`"/"`ncurses`" API.

8.1.4.2 Python 2.7.11

Python 2.7.11 represents the latest bug-fix-only release for the mature Python 2.x language generation.

On 3-Nov-2008, the Python Software Foundation issued "PEP 0373 -- Python 2.7 Release Schedule".

"Abstract

This document describes the development and release schedule for Python 2.7. The schedule primarily concerns itself with PEP-sized items. Small features may be added up to and including the first beta release. Bugs may be fixed until the final release.

Update

The End Of Life date (EOL, sunset date) for Python 2.7 has been moved five years into the future, to 2020.

This decision was made to clarify the status of Python 2.7 and relieve worries for those users who cannot yet migrate to Python 3. See also PEP 466 .

This declaration does not guarantee that bugfix releases will be made on a regular basis, but it should enable volunteers who want to contribute bugfixes for Python 2.7 and it should satisfy vendors who still have to support Python 2 for years to come.

There will be no Python 2.8 (see PEP 404)."

It would require considerable research and expertise to design, implement and debug the extensive changes to the Python curses library module needed to unofficially support the complete "curses"/"ncurses" API.

8.1.4.3 Python 3.6.0, Alpha 1

Python 3.6.0 represents the latest feature enhancement and bug-fix release for the evolving Python 3.x language generation.

8.1.4.4 Third-Party Alternatives to "curses"/"ncurses"

If there were sufficient toolkit user need, the TeamSTARS "tsWxGTUI_PyVx" Toolkit might be redesigned to support Third Party alternatives to "curses" / "ncurses", such as:

1 "PDCurses", a public-domain "Curses" programming library.

It is implemented in the "C" programming language for "Microsoft DOS", "IBM OS/2", "Microsoft Windows", "X11" and "Simple DirectMedia Layer" (SDL).

It is currently designed for 32-bit processors but will also run on 64-bit processors,

To provide functionality on "Microsoft Windows", "PDCurses" requires manual installation of the "PDCurses" Dynamic Link Library for SDL.

It would require considerable research and expertise to design, implement and debug an unofficial PDCurses API alternative to the existing Python curses library module.

It might require less research and expertise to use a Simplified Wrapper and Interface Generator (SWIG) to design, implement and debug an PDCurses API alternative to the existing Python curses library module. SWIG is an open-source software tool used to connect computer programs or libraries written in C or C++ with scripting languages such as Python. The resulting API would likely be different enough from the Python curses library module API to require changes to the TeamSTARS "tsWxGTUI_PyVx" Toolkit and to application programs which use the toolkit.

2 "UniCurses", a wrapper for Python 2.x and Python 3.x.

It provides a unified set of "Curses" functions on all platforms ("GNU/Linux", "Mac OS X", "Microsoft Windows" and "Unix") with syntax close to that of the original "Curses" / "NCurses".

It is currently designed for 32-bit processors but will also run on 64-bit processors,

To provide functionality on "Microsoft Windows", "UniCurses" wraps "PDCurses". "PDCurses" requires manual installation of the "PDCurses" Dynamic Link Library for SDL.

8.1.5 Python Application Program Launch Considerations

Python Application Programs must be launched via the Command Line User Interface (CLI) because they cannot use the Python curses library module until it has been imported, initialized, checked for terminal support of color features and the application has then started the color subsystem.

When launched via the Command Line User Interface (CLI), Python application programs can receive optional application specific key word-value pairs and positional arguments.

8.1.6 TeamSTARS "tsWxGTUI_PyVx" Toolkit Considerations

The Application Programming Interface (API) of the TeamSTARS "tsWxGTUI_PyVx" Toolkit emulates only a text-mode subset of the pixel-mode "wxPython" Graphical User Interface (GUI). The emulation uses text and graphic characters to create displays on text-mode terminals and terminal emulators having the following features:

1 Display Output

The color (xterm-family) and non-color (vt100-family) displays may contain horizontal and vertical lines, side-by-side and overlapping windows with or without titles, window size and termination control buttons, menu bars, tool bars, scroll bars, status bars, task bars, buttons, checkboxes, radio boxes & buttons, gauges, date and time stamped event messages, text with or without color and intensity markup, and other GUI objects. Displays may be laid out with or without the help of box and grid sizer services.

On August 8, 2015, Thomas E. Dickey (the long serving maintainer of ncurses) announced the release of ncurses 6.0:

"The ncurses (new curses) library is a free software emulation of curses in System V Release 4.0 (SVr4), and more. It uses terminfo format, supports pads and color and multiple highlights and forms characters and function-key mapping, and has all the other SVr4-curses enhancements over BSD curses. SVr4 curses is better known today as X/Open Curses."

The ncurses 6.0 API and Application Binary Interface (ABI) provides:

- a) backward compatible with ncurses 5.0-5.9 on 32-bit processors
- b) backward compatible with ncurses 5.0-5.9 on 64-bit processors
- c) new features with ncurses 6.0-6.x on 64-bit processors include support for a wheel mouse and for more than 16 colors and 256 color pairs.

2 Keyboard Input

3 Mouse, Trackball, Touchpad or Touchscreen Input

Future releases of the TeamSTARS "tsWxGTUI_Py_Vx" Toolkit ought to support both the API and ABI for ncurses 5.0-5.9 and for ncurses 6.0.

8.1.7 Reference Documentation

1 cpython 3.6.0 alpha 1:

<https://docs.python.org/devguide/>

2 ncurses 5.0-5.9:

<http://invisible-island.net/ncurses/ncurses.html>

3 ncurses 6.0:

<http://invisible-island.net/ncurses/announce.html>

4 pdcurses 3.4:

<https://github.com/wmcbrine/PDCurses>

5 unicurses 1.2:

<https://github.com/Chiel92/unicurses>

Draft

9 APPENDIXES

Appendixes may be used to provide information published separately for convenience in document maintenance (e.g., charts, classified data). As applicable, each appendix shall be referenced in the main body of the document where the data would normally have been provided. Appendixes may be bound as separate documents for ease in handling. Appendixes shall be lettered alphabetically (A, B, etc.).

Draft

Draft

10 TO-DO-LIST

Track and plan changes to "tsWxPkg" source files located in:

- 1 /WR/SoftwareGadgetry-PyPI
 - a) Python-2x/tsWxGTUI_Py2x
 - tsLibCLI
 - tsLibGUI
 - tsToolsCLI
 - tsToolsGUI
 - tsUtilities
 - b) Python-3x/tsWxGTUI_Py3x
 - tsLibCLI
 - tsLibGUI
 - tsToolsCLI
 - tsToolsGUI
 - tsUtilities

10.1 New Features:

- 1 Determine functional and interface requirements for determining and displaying the label character of the checkbox, radiobox or other button to be highlighted.
- 2 Modify accelerator related modules to save a separate table in each top-level window (i.e., earliest ancestor) so that the control remains local to each top-level window rather than shared among all. This will be needed to distinguish the window sizing buttons of each top-level-window.
- 3 For those platforms lacking mouse input capability, mark (with bold or standout highlights) each accelerator character that the operator should enter, via the keyboard, to select and control the associated checkbox, radiobox and other button.
- 4 Identify, dispatch and handle events associated with keyboard and mouse inputs. wxPython provides a flexible and user extensible dispatching mechanism that has not been emulated by the "tsWxGTUI_PyVx" Toolkit. It currently uses customized callbacks suited to scrolling of text windows and customized methods for the setting/clearing of check boxes and radio buttons.

10.2 Modifications:

- 1 Replace application references to curses's eight color-number based Foreground/Background colors with references to curses's eight color names.
 - Completed design, code, unit test, integration test and system test.
- 2 On eight-color cygwin, xterm and xterm-color terminals, map wxPython's sixty-eight color-names to curses's eight color-names.
 - Completed design, code, unit test, integration test and system test.
- 3 On 16-color xterm-16color terminals, map wxPython's sixty-eight and three additional xterm-16color names to curses's 16 color-names.
 - Completed design, code, unit test, integration test and system test.
- 4 On xterm-88color terminals, define color palette and color pairs to support wxPython's sixty-eight and three additional xterm-16color names.
 - Completed design, code, unit test, integration test but failed system test because host computer's standard nCurses library and Python 's curses module, were NOT built to support 256-color palette.
 - CentOS 7.0 applies 256 color-pair limit to xterm-256color emulator which is equivalent to mandating xterm-16color palette. This might be the final workaround.
- 5 On xterm-256color terminals, define color palette and color pairs to support wxPython's sixty-eight, three additional xterm-16color and an arbitrary selection of 69 additional color names.
 - Completed design, code, unit test, integration test but failed system test because host computer's standard nCurses library and Python 's curses module, weres NOT built to support 256-color palette.
 - CentOS 7.0 applies 256 color-pair limit to xterm-256color emulator which is equivalent to mandating xterm-16color palette. This might be the final workaround.
- 6 Enhance design to take advantage of curses.panels capabilities in order to support focus shifts between overlaying windows..
 - Under Construction.

10.3 Troubleshooting:

- 1 Debug tsWxPkg failure to report character under caret when over static text for title and status bar of a frame.
 - 05/31/2010 Exception raised by mouse click on any non-task bar window. Buttons are examples of failure triggers.
 - 04/15/2010 Code inspection and wing trace of tsWxStatusbar.py revealed no error in establishment of EarliestAncestor.

- 2 Debug txWxPkg failure to restore terminal to pre-startup state. For example, ctrl-C on Cygwin and Mac OS X Xterm should not result in display of ^C.
 - 05/31/2010 Failure persists despite incorporation of code to perform "stty sane".
 - 04/16/2010, reverted tsWxGraphicalTextUserInterface.py to place echo statements as in version 270-dated 06/28/2009 of tsGraphicalTextUserInterface.py because the ctrl-C problem did not exist in the earlier urwid-based design. This did not fix the problem.
- 3 Debug incorrect color pair displays for xterm-256color. Frame displayed with dark green instead of blue. Dialog displayed with darker green instead of cyan. Stand-alone test of tsWxGraphicalTextUserInterface has not verified color bars due to short display time (2-seconds) and lack of reference colors.
 - Google Search: NCurses 256 Color Support: But note that some terminals, while they can support 256 colors, are not able to change the palette. To compile NCurses with 256 color support, ... www.c-for-dummies.com/nCurses/256color/
 - If Python uses standard nCurses, it would only support 8 colors.
 - Must therefore suspend effort to resolve this issue.
 - CentOS 7.0 applies 256 color-pair limit to xterm-256color emulator which is equivalent to mandating xterm-16color palette. This might be the final workaround.
- 4 Debug compressed "ansi" display. Determine if due to Cygwin and Mac OS X ansi emulator. By contrast, Mac OS X vt100 emulator does work.
 - On Mac OS X, the "Terminal" utility creates a almost normal ansi display using "?" for vertical lines and "q" for horizontal lines. However, the "iTerm" application is unable to even create the shapes of the windows.
 - Must therefore suspend effort to resolve this issue.
- 5 Debug scrolling vt100 display. Determine if due to Cygwin vt100 emulator. By contrast, Cygwin and Mac OS X vt100 emulators in xterm window do work.
 - On Windows, the "Cygwin XWin Server" utility creates a normal display but "Cygwin Bash Shell" utility does not. Each screen refresh overflows the "Redirected Output" window beyond its top border leaving multiple copies of the top two lines visible.
 - Must therefore suspend effort to resolve this issue.

10.4 Validation/Regression Test:

- 1 No Validation and Regression testing of the following terminal emulations and database classes:
 - a) 8-color ansi terminal emulator with and without mapping from 68-color wxPython palette because no host platform reproduces GUI object shapes and border lines
- 2 Validation and Regression testing of the following terminal emulations and associated database classes:
 - a) Non-color vt100 and vt220 terminal emulators

- b) 8-color cygwin mintty, xterm and xterm-color with and without mapping from standard 68-color wxPython palette
 - c) 16-color xterm-16color terminal emulator with and without mapping from enhanced 71-color wxPython-style palette
 - d) 16-color xterm-88color terminal emulator with and without mapping from enhanced 71-color wxPython-style palette
 - e) 16-color xterm-256color terminal emulator with and without mapping from enhanced 140-color wxPython-style palette
- 3** Validate establishment of class parent, child, earliest ancestor and curses panel GUI object overlay level relationships for each task.

Draft

11 SAMPLE SCREEN SHOTS

From Wikipedia, the free encyclopedia

"A screen dump, screen capture (or screen-cap), screenshot (or screen shot), screengrab (or screen grab), or print screen[1] is an image taken by the computer user to record the visible items displayed on the monitor, television, or another visual output device. Usually, this is a digital image using the (host) operating system or software running on the computer, but it can also be a capture made by a camera or a device intercepting the video output of the display (such as a DVR). That latent image converted and saved to an image file such as to JPEG or PNG format is also called a screenshot.

Screenshots can be used to demonstrate a program, a particular problem a user might be having, or generally when display output needs to be shown to others or archived. For example, after being emailed a screenshot, a Web page author might be surprised to see how his page looks on a different Web browser and can take corrective action. Likewise with differing email software programs, (particularly such as in a cell phone, tablet, etc.,) a sender might have no idea how his email looks to others until he sees a screenshot from another computer and can (hopefully) tweak his settings appropriately."

NOTES:

1) The Python "Curses" module interfaces with the local platform's "nCurses" programming library which then interfaces with the platform's terminal emulator software and windowing system. For example, Microsoft provides the Power Windows console, Cygwin, Mac OS X and various Linux and UNIX distributions provide a Bash shell and an X11-based terminal emulator. Cygwin also provides an X11-based mintty console.

2) Some X11-base terminal emulators render window borders with solid lines; others render borders with dotted lines. Color renditions also vary from platform to platform.

3) Some vt100/vt220 terminal emulations augment the single foreground color with a second color for highlighting. None generate XTERM compatible mouse-click events. Some block mouse-click event while others generate a sequence of non-standard mouse-click events.

11.1 test_tsWxLinesOfCode

Application using xterm via Terminal on Mac OS X

NOTE: This is a demonstration of the adaptation of a command line application that uses a TextCtrl widget to append its output to the blue window rather than just using print statements which would scroll out of sight in the green Redirected Output window.

```
Prototype_Feature
File

test_tsWxLinesOfCode, v1.3.0 (build 12/25/2011)

Authors: Richard S. Gordon, a.k.a. Software Gadgetry
Copyright (c) 2007-2011 Software Gadgetry. All rights reserved.

-----

Input Path:  <./>.
Results are available in "/Users/rsg/WR/SoftwareGadgetry-2.x/tsWxLinesOfCodeStatistics.txt".

-----

      FILES      CODE      CMNTS      LINES      WORDS      CHARS
Pct:           54.00%    45.91%    100.00%
Totals:    1296    274409    232913    507322    1519906    15805645
Std:       1185      411      379      742      2298      25582
Avg:         1       212      180      392      1173      12196

-----

Skipped 2509 of 3805 file(s) for having invalid "name.ext":

Valid file "name.ext" (Upper or Lower case):

.cda, .asm, .bas, .bash, .bat, .c, .c++, .cpp,
.csh, .f, .f77, .f90, .for, .ftn, .g77, .g90,
.h, .inc, .ksh, .pas, .plm, .py, .sh

-----

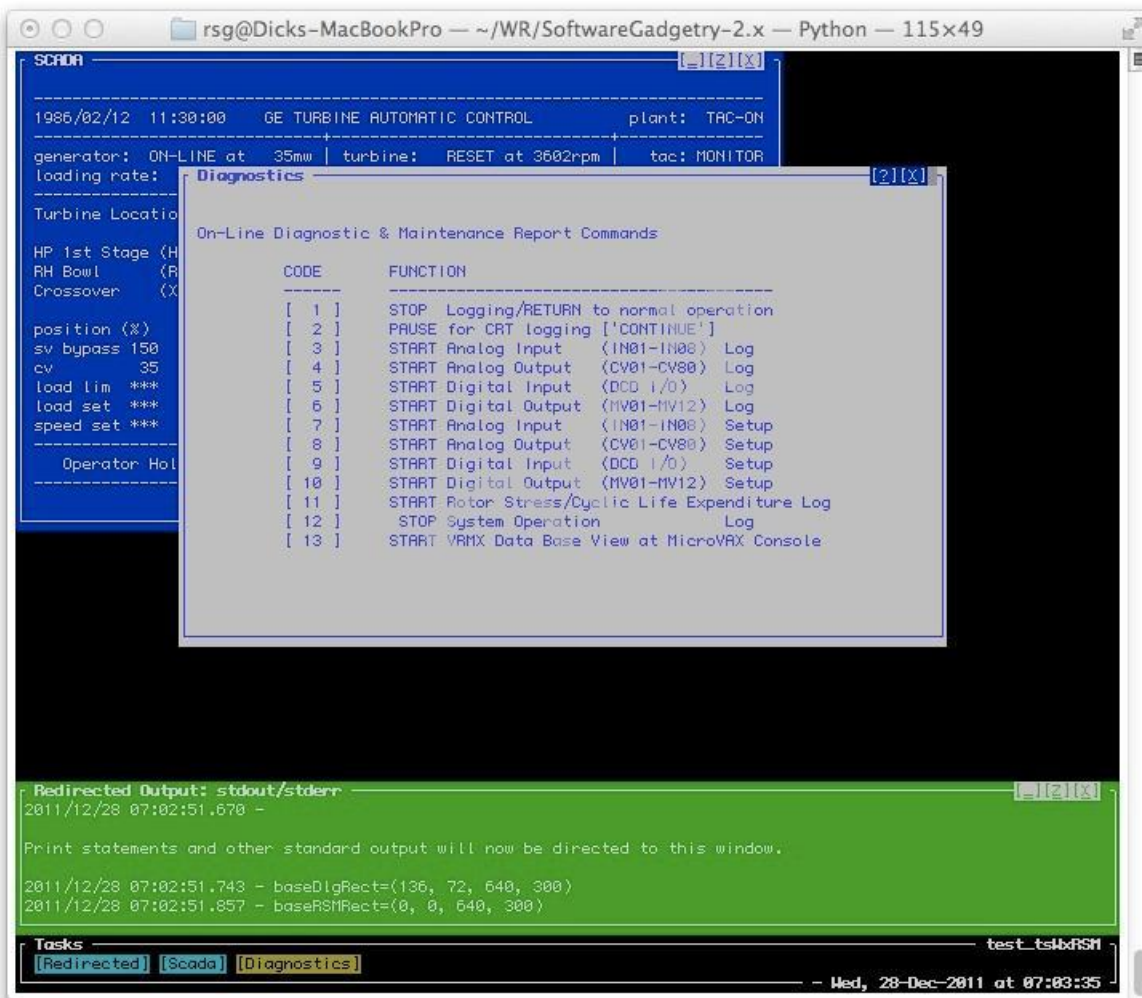
Redirected Output: stdout/stderr

2011/12/28 07:05:42.106 - Results are available in "/Users/rsg/WR/SoftwareGadgetry-2.x/tsWxLinesOfCodeStatistics.txt".

Tasks
[Redirected] [Prototype_Feature]
/ Wed, 28-Dec-2011 at 07:07:12
```

11.2 test_tsWxRSM Application using xterm via Terminal on Mac OS X

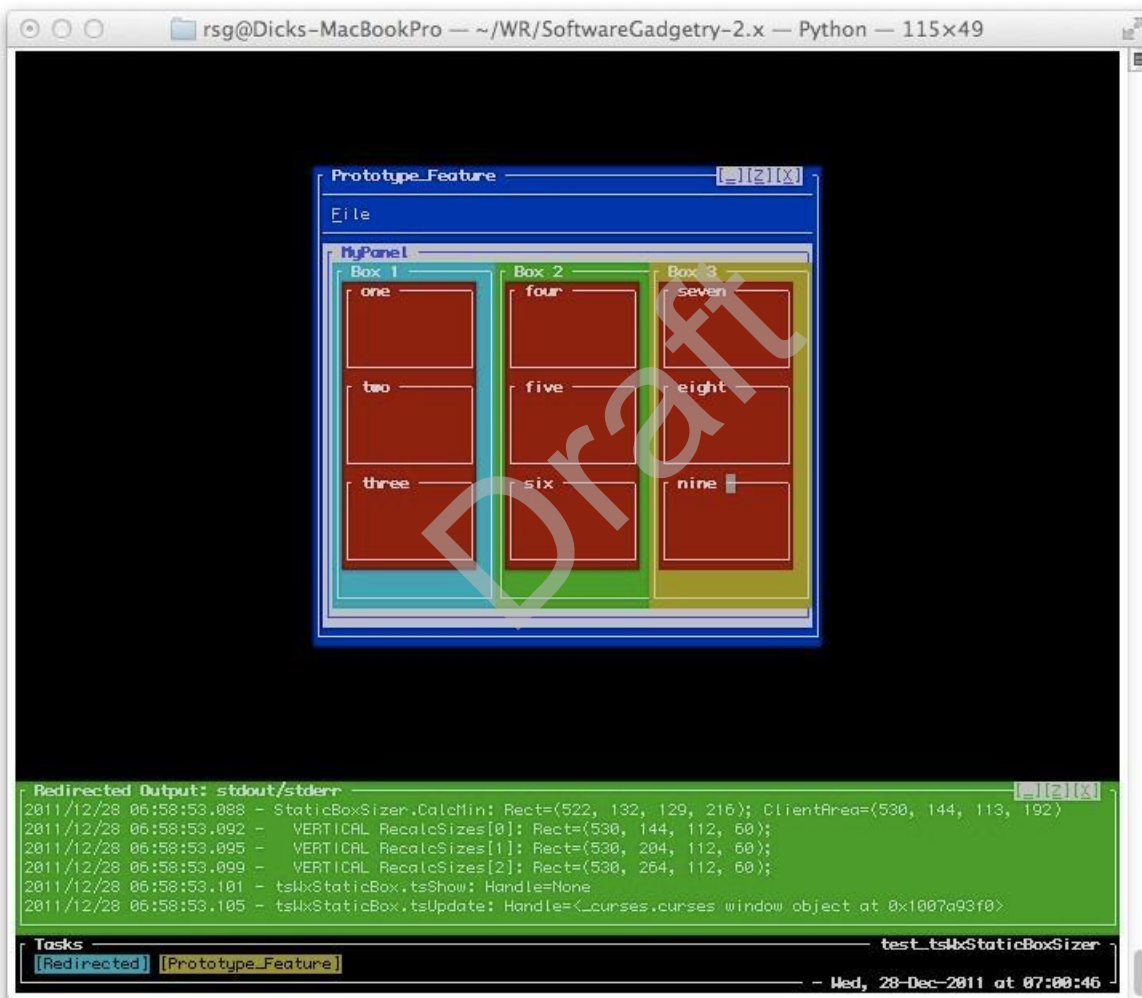
NOTE: This is a simulation of an instrumentation and diagnostic application that uses a TextCtrl widget to concurrently append its output to the blue and white windows rather than just updating selected fields in each window when the underlying instrumentation or diagnostic data changed.



11.3test_tsWxStaticBoxSizer

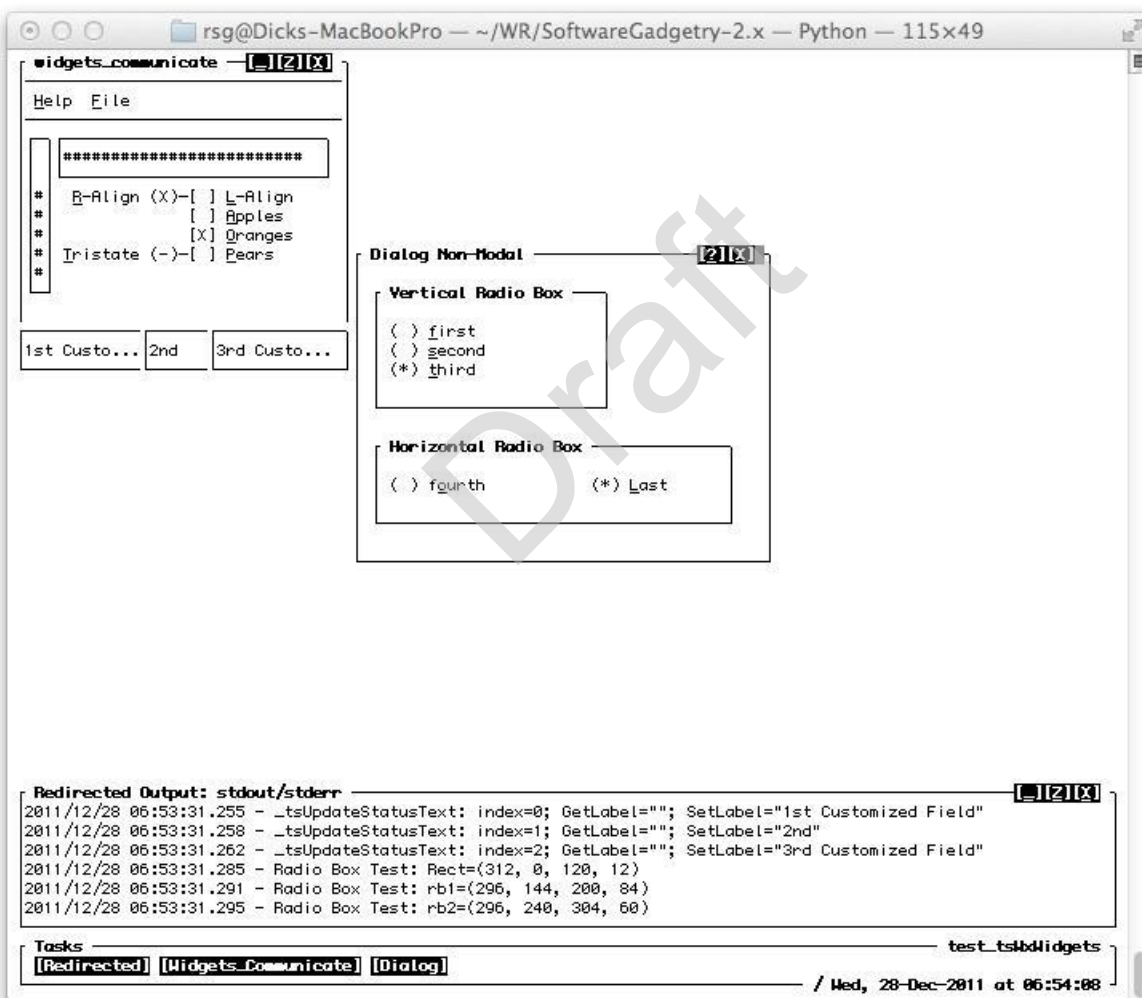
Application using xterm via Terminal on Mac OS X

NOTE: This is a demonstartion of the BoxSizer widget operation.



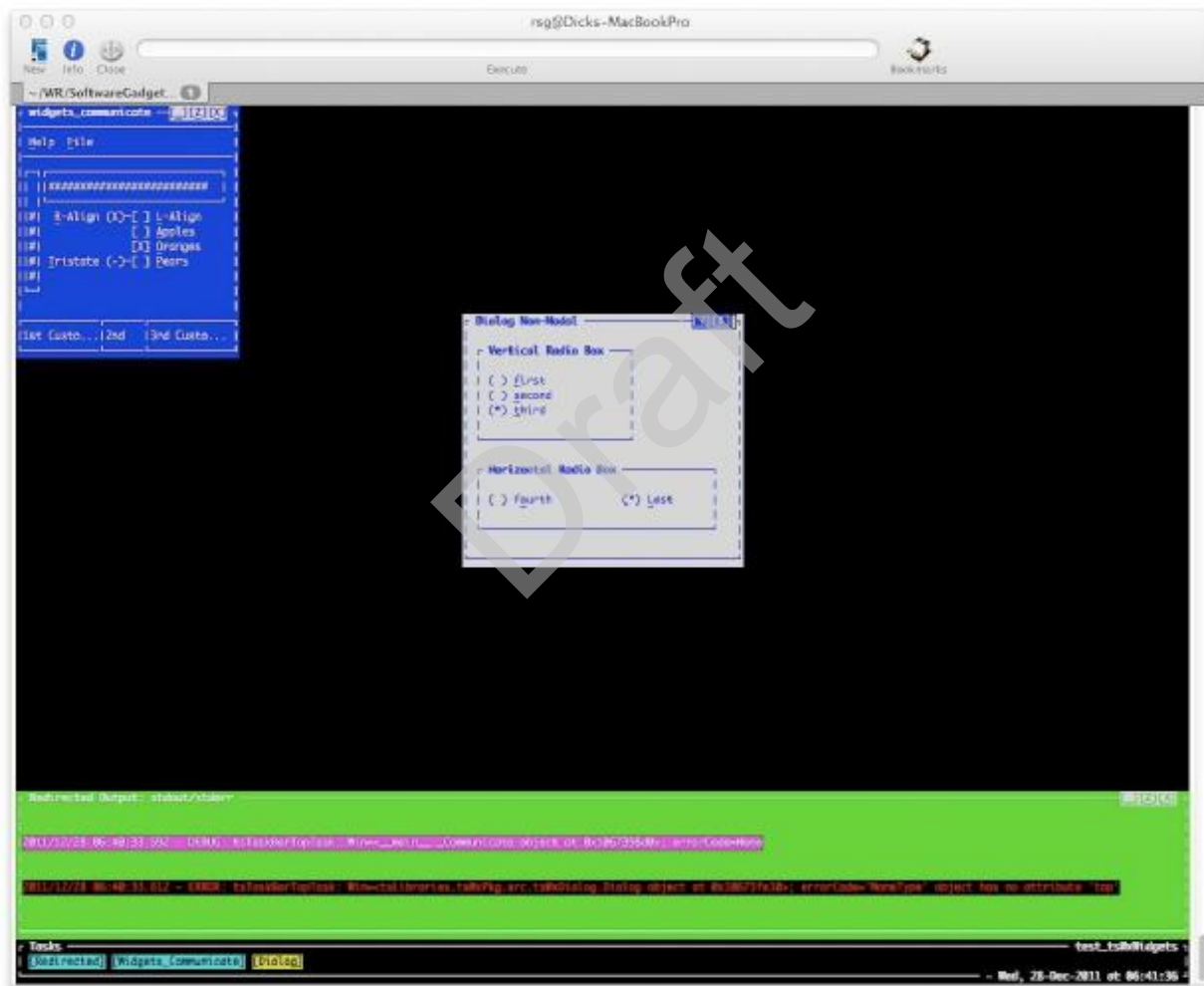
11.4 test_tsWxWidgets Application using vt100 via Terminal on Mac OS X

NOTE: This is a demonstration of various widgets on a terminal that reports NOT having colors. It is the same application that runs on terminals that reports having colors. The application remains unaware of the presence or absence of colors.



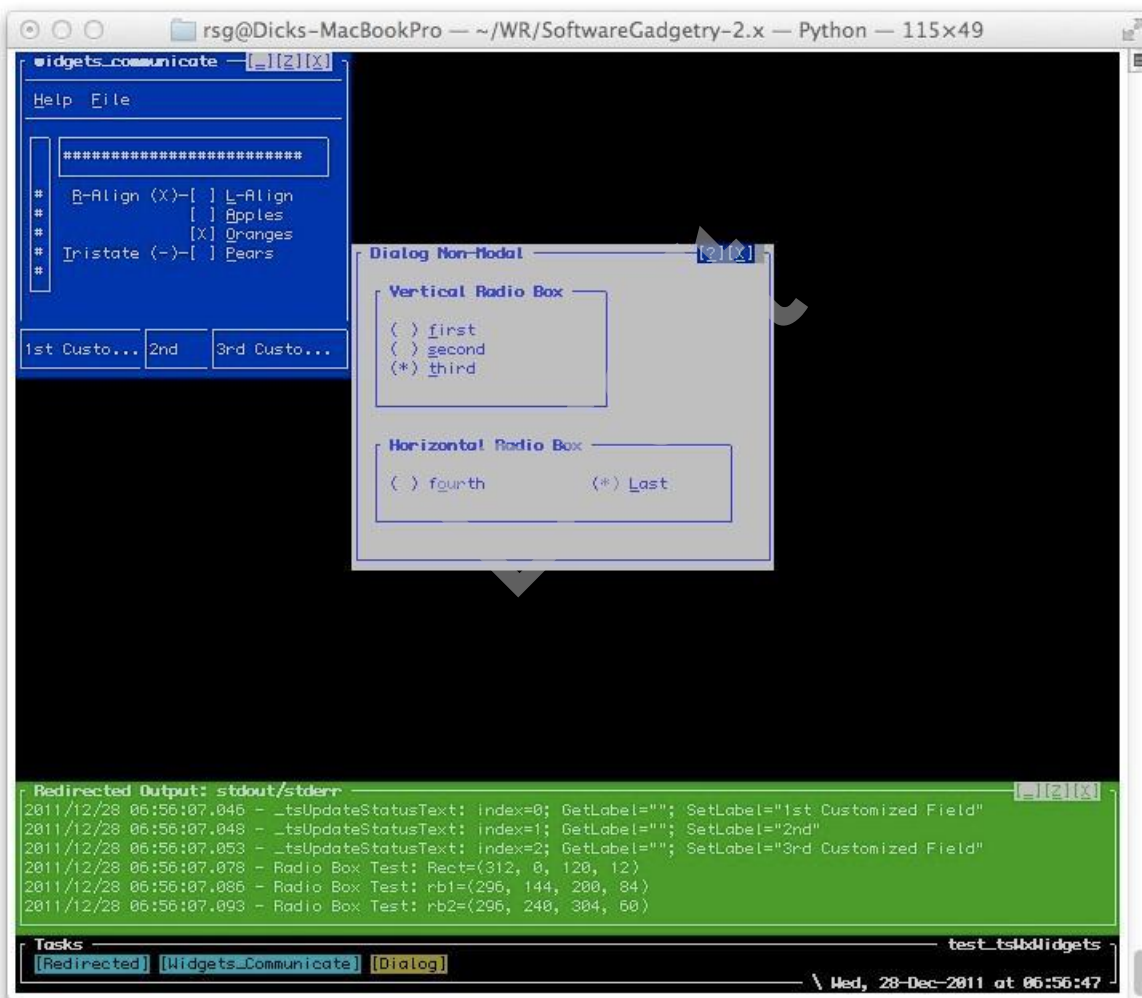
11.5test_tsWxWidgets Application using xterm via iTerm on Mac OS X

NOTE: This is a demonstration of various widgets and the use of "DEBUG:" and "ERROR:" markups in Redired Output messages.



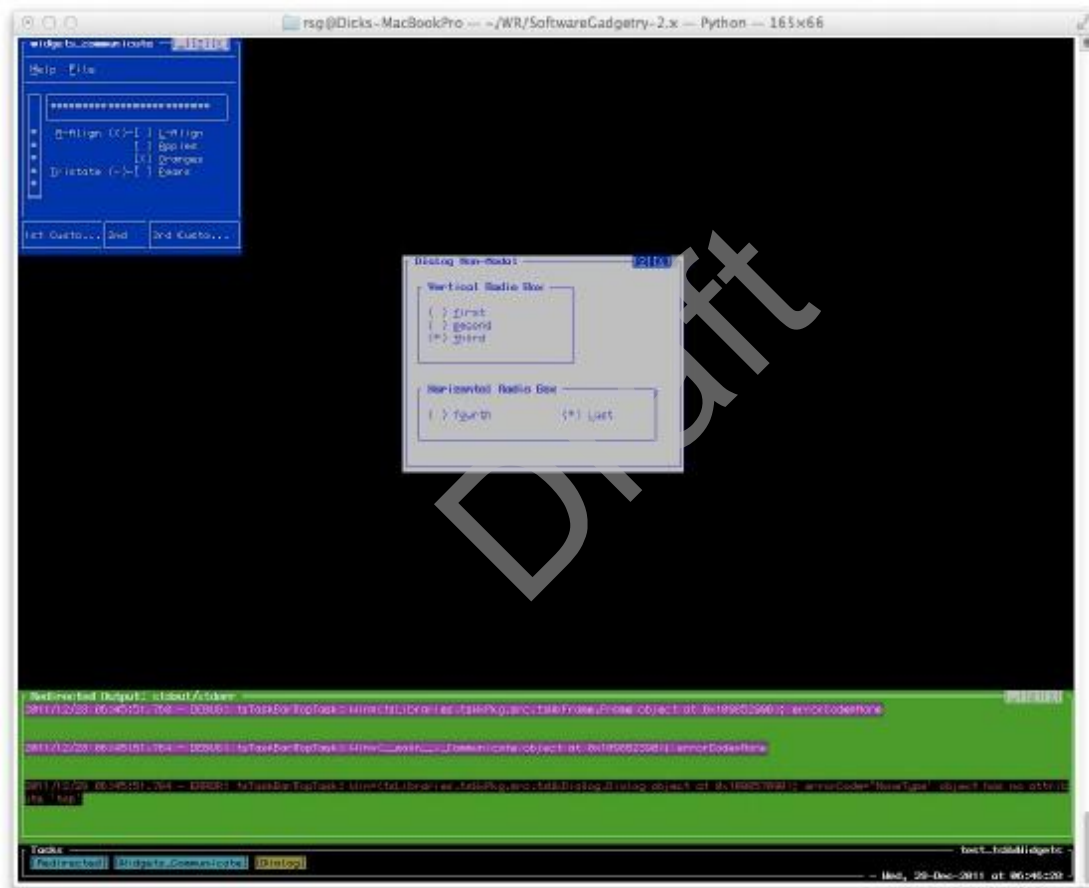
11.6 test_tsWxWidgets Application using xterm via Terminal on Mac OS X

NOTE: This is a demonstration of various widgets and the use of Redired Output messages lacking markups such as "DEBUG:" and "ERROR:." .



11.7 test_tsWxWidgets Application using xterm via Terminal on Mac OS X

NOTE: This is a demonstration of various widgets, after startup in a larger screen, and the use of "DEBUG:" and "ERROR:" markings in Redired Output messages.



11.8test_tsWxMarkupDiagnostics via Cygwin-Mintty Terminal on Mac OS X

NOTE: This is a demonstration of various markups, after startup in a larger screen. The markups are overlaid on existing windows to assess their visibility against different foreground and background combinations.

