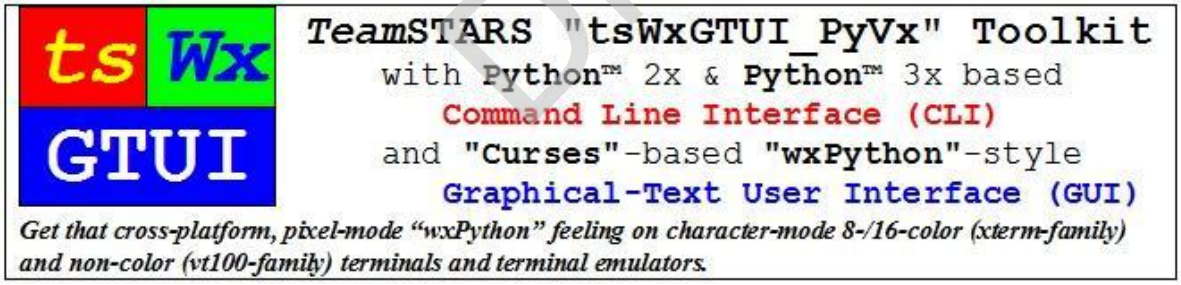
Software Gadgetry

Use Cases

Vol. 13 - "tsWxGTUI_PyVx" Toolkit

Rev. 0.0.7 (Pre-Alpha)

Author(s): Richard S. Gordon



Printed on 25 August, 2016

Contents

1 Computer User Use Case(s)

From Wikipedia, the free encyclopedia

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

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

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

On-line documentation for the *Team*STARS "tsWxGTUI_PyVx" Toolkit is provided in the subdirectory named:

```
"./<Toolkit Recipient's Repository>/Documents".

<Your Working Repository>
(e.g. "tsWxGTUI_PyVx_Repository")
  |
  |
  +-- ["Documents"]
  |      |
  |      |  This directory contains a collection of files
  |      |  which provide the Toolkit recipient with an
  |      |  understanding of the purpose, goals & capabil-
  |      |  ities, non-goals & limitations, terms & condi-
  |      |  tions and procedures for installing, operating,
  |      |  modifying and redistributing the Toolkit.
  |
  +-- ["ManPages"]
  |      |
  |      |  Deliverable Toolkit manual pages are 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.
  |
```

```
+-- ["Notebooks"]
|      |
|      |  Contains a collection of commentaries that
|      |  express opinions or offerings of explana-
|      |  tions about events or situations that might
|      |  be useful to Toolkit installers, developers,
|      |  operators, troubleshooters and distributors.
|      |  The documents may be in Application-specific
|      |  formats (such as Adobe PDF, JPEG Bit-mapped
|      |  image, LibreOffice, Microsoft Office, plain
|      |  text).
|
+-- ["SourceDistributions"]
|      |
|      |  Contains a collection of computer program
|      |  source code files that the Toolkit recip-
|      |  ient will need to install, operate, modify
|      |  and re-distribute the Toolkit.
|
+-- "README.txt"
|      |
|      |  Contains information about other files in
|      |  a directory or archive and is commonly
|      |  distributed with computer software, forming
|      |  part of its documentation.
|
+-- "MANIFEST.txt"
       |
       |  Contains a Tally List for deliverable items.
```

1.1 Role-Specific Use Cases

The Documents directory contains a collection of files which provide the Toolkit recipient with an understanding of the purpose, goals & capabilities, non-goals & limitation, terms & conditions and procedures and for installing, operating, modifying and redistributing the Toolkit.

```
"Announcement.htm" --- Introduces prospective and new
"Announcement.pdf"     recipients to the purpose, goals,
"Announcement.rtf"     non-goals, design and features of
"Announcement.txt"     this computer software. Provided
                       in single and multi-font formats.

"AUTHORS.txt"      --- List of the principal "tsWxGTUI_PyVx"
                       Toolkit author(s) and authors
                       credited for work covered by a prior
                       copyright and license.

"BUGS.txt"         --- List of Known Problems / Issues.

"CHANGE_LOG.txt"   --- List of Additions, Modification and
                       Deletions.

"CONFIGURE.txt"    --- Instructions for applying factory and
                       site-specific configurations.

"COPYING.txt"      --- Instructions for copying all or a
                       portion of the distribution.

"COPYRIGHT.txt"    --- Declaration of the exclusive legal
                       right, given to an originator or an
                       assignee to use, copy and distribute
                       computer software, and to authorize
                       others to do the same.

"CREDITS.txt"      --- Acknowledgment given to those whose
                       Copyrighted Work is used in accordance
                       with it's originator's Copyright and
                       License.

"DEMO.txt"         --- Narrated script demonstrating how to
                       install, configure, operate and trouble-
                       shoot the TeamSTARS "tsWxGTUI_PyVx"
                       Toolkit.

"FAQ.txt"          --- Answers to Frequently Asked
                       Questions.

"GETTING_STARTED.txt" --- Introduces new recipients to
                          the system requirements and
                          third-party resources avail-
                          able to new Toolkit users.
```

```
"INSTALL.txt"       --- Describes steps to download, extract
                        install and configure the "tsWxGUI"
                        Toolkit.

"LICENSE.txt"       --- General and special arrangements,
                        provisions, rules, specifications and
                        standards that form an integral part
                        of the agreement or contract between
                        the creator and recipient of Copy-
                        righted and Licensed Work.

"NEWS.txt"          --- Announcements of new releases.

"NOTICES.txt"       --- Details the copyright(s) and li-
                        cense(s).

"OPERATE.txt"       --- Describes steps to use the "tsWxGUI"
                        Toolkit.

"README.txt"        --- Introduces new recipients to the
                        purpose, goals, non-goals, design
                        and features of the computer
                        software product. Supplements
                        include the following:

                        "README1-Introduction.txt"
                        "README2-Repository.txt"
                        "README3-Documents.txt"
                        "README4-ManPages.txt"
                        "README5-Notebooks.txt"
                        "README6-SourceDistributions.txt"
                        "README7-DeveloperSandboxes.txt"
                        "README8-SitePackages.txt"
                        "README9-KeyboardMouseInput.txt"

"THANKS.txt"        --- Acknowledgments to those otherwise
                        unsung heros who contributed time
                        and effort to supporting the authors
                        as planners, editors, designers,
                        coders and testers.

"TO-DO.txt"         --- A To-Do-List provides a roadmap for
                        development and troubleshooting work.

"TROUBLESHOOT.txt" --- Provides a list of available refer-
                        ence resources and a guide for plan-
                        ning, developing and troubleshooting
                        a cross-platform system of hundreds
                        of files each containing a few, tens
                        or hundred of class, data and method
                        definitions. Its complexity becomes
                        apparent in the recent software
                        Lines-Of-Code metrics.
```


The following use case(s) describe the expected activites associated with various job roles.

1.1.1 System Administrator

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

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

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

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

1.1.2 Software Engineer

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

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

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

1.1.3 System Operator

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

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

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

At earlier stages in the computer system's installation, configuration and software development process, the System Operator may temporarily be a **System Administrator** (on page 7) or **Software Engineer** (on page 7).

At later stages in the computer system's maintenance, the System Operator may temporarily be the **Field Service Engineer (on page 8).**

The activities are application-specific, system-specific and role-specific.

The *Team*STARS "tsWxGTUI_PyVx" Toolkit application-specific use cases are detailed in the downloaded copy of the followig document:

- https://GitHub.com/rigordo959/tsWxGTUI_PyVx_Repository/Documents/DEMO.txt

The following use cases illustrate the system specific similarities and differences:

1. ***Linux Use Case*** (on page 9)
2. ***Mac OS X Use Case*** (on page 10)
3. ***Microsoft Windows Use Case(s)*** (on page 11)
   a) ***Linux-Like (Cygwin) Use Case*** (on page 11)
   b) ***DOS-Like Use Case*** (on page 12)
4. ***Unix Use Case*** (on page 13)

1.1.4 Field Service Engineer

A Field Service Engineer is a professional who is responsible for the installation of hardware and software products, or the maintenance and repair of already-installed products.

Exerpt From http://dot-job-descriptions.careerplanner.com/FIELD-SERVICE-ENGINEER.cfm

1. "Installs and repairs electronic equipment, such as computer, radar, missile-control, avionics, and communication systems, in field installations: Consults with customer or supervisor to plan layout of equipment.
2. Studies blueprints, schematics, manuals, and other specifications to determine installation procedures.

3 Installs or oversees installation of equipment according to manufacturer's specifications.

4 Operates system to demonstrate equipment and to analyze malfunctions.

5 Interprets maintenance manuals, schematics, and wiring diagrams, and repairs equipment, utilizing knowledge of electronics and using standard test instruments and handtools.

6 Instructs and directs workers in servicing and repairing equipment.

7 Consults with engineering personnel to resolve unusual problems in system operation and maintenance.

8 May instruct workers in electronic theory.

9 May supervise workers in testing, tuning, and adjusting equipment to obtain optimum operating performance.

10 May advise management regarding customer satisfaction, product performance, and suggestions for product improvements."

1.2 System-Specific Use Cases

The following use case(s) describe the expected activites associated with various hardware and operating system platforms.

1.2.1 Linux Use Case

Linux (Debian, Fedora, SuSE, Ubuntu etc.) Use Case

Operation of a typical computer system advances through various stages:

1 Manual Power-ON of Host Computer Hardware (Booting)

2 Automatic reset of Host Computer Hardware

3 Automatic startup of Host Computer Operating System Software

4 Automatic startup of Host Computer Pixel-mode Graphical User Interface Software (**login**)

5 Optional installation of new or updated Host Computer Operating System Libraries and Software

> See the file "**GETTING_STARTED.txt**" (in the "**./Documents**" directory) for platform preparation instructions.

6 Optional installation of new or updated "tsWxGTUI_PyVx" Toolkit Application Libraries and Software

7 Manual startup of Host Computer Command Line Interface Software (**Applications -> Accessories -> terminal**)

For those application programs that require manual termination, it may be necessary for the System Opertor to issue an **exit** command (via text from the Command Line Interface or via a BUTTON from the Graphical User Interface) or an **abort** command (via a CTRL-C signal from the Command Line Interface).

8 Manual shutdown of Host Computer Command Line Interface Software (**exit**)

9 Manual shutdown of Host Computer Pixel-mode Graphical User Interface Software (**System -> Logout**)

10 Manual shutdown of Host Computer Operating System Software (**System -> Shutdown**)

11 Manual Power-OFF of Host Computer Hardware (Shutdown)

1.2.2 Mac OS X Use Case

Mac OS X (10.3 -10.11) Use Case

Operation of a typical computer system advances through various stages:

1 Manual Power-ON of Host Computer Hardware (Booting)

2 Automatic reset of Host Computer Hardware

3 Automatic startup of Host Computer Operating System Software

4 Automatic startup of Host Computer Pixel-mode Graphical User Interface Software (**login**)

5 Optional installation of new or updated Host Computer Operating System Libraries and Software

See the file "**GETTING_STARTED.txt**" (in the "**./Documents**" directory) for platform preparation instructions.

6 Optional installation of new or updated "tsWxGTUI_PyVx" Toolkit Application Libraries and Software

7 Manual startup of Host Computer Command Line Interface Software (**Applications -> iTerm** or **Applications -> Utilities - > terminal**)

For those application programs that require manual termination, it may be necessary for the System Opertor to issue an **exit** command (via text from the Command Line Interface or via a BUTTON from the Graphical User Interface) or an **abort** command (via a CTRL-C signal from the Command Line Interface).

8 Manual shutdown of Host Computer Command Line Interface Software (**exit**)

9 Manual shutdown of Host Computer Pixel-mode Graphical User Interface Software (**logout**)

10 Manual shutdown of Host Computer Operating System Software (**shutdown**)

11 Manual Power-OFF of Host Computer Hardware (Shutdown)

1.2.3 Microsoft Windows Use Case(s)

Microsoft Windows (XP, 7, 8,/8.1, 10) Use Case)

NOTES:

1) "Cygwin" is a free Linux-like plug-in environment, from Red Hat, for 32-bit and 64-bit versions of Microsoft Windows. It includes a Command Line Interface (e.g. bash shell), GNU compiler and utility toolchain, Python programming language tools and the curses Terminal Control library.

2) Without the "Cygwin" plug-in, a Microsoft Windows system will not be able to operate the *Team*STARS "tsWxGTUI_PyVx" Toolkit in its Graphical User Interface (GUI) mode.

3) Without the Python programming language tools, a Microsoft Windows system will not be able to execute any of the *Team*STARS "tsWxGTUI_PyVx" Toolkit software.

Follow the procedure appropriate to your need for and availability of "Cygwin":

1 ***Linux-Like (Cygwin) Use Case*** (on page 11)

2 ***DOS-Like Use Case*** (on page 12)

1.2.3.1 Linux-Like (Cygwin) Use Case

Linux-Like (Cygwin 1.7.8 - 2.5.1) Use Case for Microsoft Windows (XP, 7, 8,/8.1, 10)

Use this procedure only AFTER the installation of "Cygwin".

Operation of a typical computer system advances through various stages:

1 Manual Power-ON of Host Computer Hardware (Booting)

2 Automatic reset of Host Computer Hardware

3 Automatic startup of Host Computer Operating System Software

4 Automatic startup of Host Computer Pixel-mode Graphical User Interface Software (**login**)

5 Optional installation of new or updated Host Computer Operating System Libraries and Software

   See the file "**GETTING_STARTED.txt**" (in the "**./Documents**" directory) for platform preparation instructions.

6 Optional installation of new or updated "tsWxGTUI_PyVx" Toolkit Application Libraries and Software

7 Manual startup of Host Computer Command Line Interface Software (**Start -> Programs -> Cygwin -> Cygwin Bash Shell**)

For those application programs that require manual termination, it may be necessary for the System Opertor to issue an **exit** command (via text from the Command Line Interface or via a BUTTON from the Graphical User Interface) or an **abort** command (via a CTRL-C signal from the Command Line Interface).

8 Manual shutdown of Host Computer Command Line Interface Software (**exit**)

9 Manual shutdown of Host Computer Pixel-mode Graphical User Interface Software (**logout**)

10 Manual shutdown of Host Computer Operating System Software (**shutdown**)

11 Manual Power-OFF of Host Computer Hardware (Shutdown)

1.2.3.2 DOS-Like Use Case

DOS-Like Use Case for Microsoft Windows (XP, 7, 8,/8.1, 10)

Use this procedure only BEFORE the installation of "Cygwin".

Operation of a typical computer system advances through various stages:

1 Manual Power-ON of Host Computer Hardware (Booting)

2 Automatic reset of Host Computer Hardware

3 Automatic startup of Host Computer Operating System Software

4 Automatic startup of Host Computer Pixel-mode Graphical User Interface Software (**login**)

5 Optional installation of new or updated Host Computer Operating System Libraries and Python Software.

See the file "**GETTING_STARTED.txt**" (in the "**./Documents**" directory) for Do-It-Yourself Python Software installation instructions.

6 Optional installation of new or updated "tsWxGTUI_PyVx" Toolkit Application Libraries and Software

7 Manual startup of Host Computer Command Line Interface Software (**Start -> Programs -> Accessories -> Command Prompt**)

For those application programs that require manual termination, it may be necessary for the System Opertor to issue an **exit** command (via text from the Command Line Interface or via a BUTTON from the Graphical User Interface) or an **abort** command (via a CTRL-C signal from the Command Line Interface).

8 Manual shutdown of Host Computer Command Line Interface Software (**exit**)

9 Manual shutdown of Host Computer Pixel-mode Graphical User Interface Software (**logout**)

10 Manual shutdown of Host Computer Operating System Software (**shutdown**)

11 Manual Power-OFF of Host Computer Hardware (Shutdown)

1.2.4 Unix Use Case

Unix (FreeBSD, PC-BSD etc.) Use Case

Operation of a typical computer system advances through various stages:

1. Manual Power-ON of Host Computer Hardware (Booting)
2. Automatic reset of Host Computer Hardware
3. Automatic startup of Host Computer Operating System Software
4. Automatic startup of Host Computer Pixel-mode Graphical User Interface Software (**login**)
5. Optional installation of new or updated Host Computer Operating System Libraries and Software

   See the file "**GETTING_STARTED.txt**" (in the "**./Documents**" directory) for platform preparation instructions.

6. Optional installation of new or updated "tsWxGTUI_PyVx" Toolkit Application Libraries and Software
7. Manual startup of Host Computer Command Line Interface Software (**Kickoff Application Launcher -> terminal**)

   For those application programs that require manual termination, it may be necessary for the System Opertor to issue an **exit** command (via text from the Command Line Interface or via a BUTTON from the Graphical User Interface) or an **abort** command (via a CTRL-C signal from the Command Line Interface).

8. Manual shutdown of Host Computer Command Line Interface Software (**exit**)
9. Manual shutdown of Host Computer Operating System Software (**Kickoff Application Launcher -> Leave-> Shutdown**)
10. Manual Power-OFF of Host Computer Hardware (Shutdown)

1.3 Application-Specific Use Cases

The following use case(s) describe the expected activites associated with various *Team*STARS "tsWxGTUI_PyVx" Toolkit Command Line Interface and Graphical User Interface applications programs.

See the file "**DEMO.txt**" (in the "**./Documents**" directory) for Application-Specific instructions:

TABLE OF CONTENTS (sample)

5. Prepare SSH / SFTP Client & Server for Remote Login
  5.1 Linux SSH / SFTP Server (CentOS 7)
  5.2 Mac OS X SSH / SFTP Server (Yosemite)
  5.3 Microsoft Windows SSH / SFTP Client & Server
    (Professional Edition of Windows 7)
    5.3.1 Cygwin Plug-In Mode (Not Verified)
    5.3.2 Native Mode (Not Verified)
  5.4 Unix SSH / SFTP Server (PC-BSD 10) (Not Verified)

6. Use SSH / SFTP Server(s) for Remote Login
  6.1 Create Remote Single Session Desktop
    6.1.1 Launch the Command Line Interface shell session.
    6.1.2 Login to Remote System as user with/without administrative privileges.
    6.1.3 Set Working Directory
    6.1.4 Launch First Application
    6.1.5 Launch Last Application
    6.1.6 Create Remote archive of logs directory
    6.1.7 Transferring files between Local and Remote Systems
    6.1.8 Logout of Remote System

7. Use Multiple Sessions for Concurrent Applications
  7.1 Create First Single/Multi-Session Desktop
    7.1.1 Launch the Command Line Interface shell session.
    7.1.2 Login to Remote System as user with/without administrative privileges.
    7.1.3 Set Working Directory
    7.1.4 Launch First Application
    7.1.5 Launch Last Application
  7.2 Create Last Single/Multi-Session Desktop
    7.2.1 Launch the Command Line Interface shell session.
    7.2.2 Login to remote system as user with/without administrative privileges.
    7.2.3 Set Working Directory
    7.2.4 Launch First Application
    7.2.5 Launch Last Application
  7.3 Logout of Single/Multi-Session Desktops
    7.3.1 Logout of First Remote System
    7.3.2 Logout of Last Remote System

2 Computer Source Code Use Case(s)

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

1 **High Level API** - A programming interface that is the least detailed but provides more functionality within one statement than a lower-level interface. High-level interfaces are designed to enable the programmer to write code in a shorter amount of time and to be less involved with the details of the software module or hardware device that is performing the required services. For example, the programmer can invoke a high level procedure to append one or more text strings to an internal buffer.

2 **Low Level API** - A programming interface that is the most detailed, allowing the programmer to manipulate functions within a software module or within hardware at a very granular level. To continue with the afore mentioned example, a high level procedure can then invoke one or more low level procedures that will sequentially get one text string at a time from the internal buffer, sequentially scroll the top most string off the display, then scroll up each remaining displayed string and finally outputting the new string to the bottom row of the screen.

3 **Extended API** - A customized version of an existing programming interface that supports additional keyword value pairs and positional arguments. For example, the "tsWxGTUI_PyVx" Toolkit is a character-mode emulation of the pixel-mode "wxPython" API. It internally may require optional parameters to distinguish such features as character-mode dimensions from their pixel-mode counterparts. It may also include functionality that "wxPython" and its underlying "wxWidgets" toolkit otherwise obtain from the host operating system and its programming libraries, such as the installation of the color palette and the implementation of scrollable windows.

2.1 Comparison with “wxPython"

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 *Team*STARS "tsWxGTUI_PyVx" Toolkit.

Feature	"wxPython" / "wxWidgets" / "wxEmbedded"	TeamSTARS "tsWxGTUI_PyVx" Toolkit
Platform	Locally (via system console or xterm) and remotely (via vnc) accessible systems: ▪ Desktop ▪ Laptop ▪ Workstation ▪ Embedded (via KOAN's "wxEmbedded")	Locally (via system console or xterm) and remotely (via xterm) accessible systems: ▪ Desktop ▪ Laptop ▪ Workstation ▪ Embedded
Operator Desktop	▪ Keyboard	▪ Keyboard

Feature	"wxPython" / "wxWidgets" / "wxEmbedded"	TeamSTARS "tsWxGTUI_PyVx" Toolkit
Platform	Locally (via system console or xterm) and remotely (via vnc) accessible systems: ▪ Desktop ▪ Laptop ▪ Workstation ▪ Embedded (via KOAN's "wxEmbedded")	Locally (via system console or xterm) and remotely (via xterm) accessible systems: ▪ Desktop ▪ Laptop ▪ Workstation ▪ Embedded
Interface Hardware	▪ Mouse, trackball, touchpad or touchscreen ▪ Optional Mouseless ▪ Display (68-color pixel mode)	▪ Mouse, trackball, touchpad or touchscreen ▪ Keyboard Shortcut Key and/or Optional Mouse (Future) ▪ Display (68-color character mode mapped into: 8-color/64-color pair on cygwin, linux, xtem or xterm-color terminals) ▪ Display (71-color character mode mapped into: 16-color/256-color pair on xterm-16color, xterm-88color or xterm-256color terminals) ▪ Display (a single shade of white, green or orange that is "ON" depending on the single available phosphor or black when "OFF" on vt100/vt220 terminals)
Operating System & Device Driver Software	▪ GNU/Linux (Linux Operating System Kernel with Unix-like GNU Command Line Interface, Graphical User Interface and Toolchain). ▪ Mac OS X (Darwin-/BSD-based Unix Operating System Kernel with GNU Command Line Interface, Toolchain and Apple Computer's Graphical User Interface and Toolchain). ▪ Microsoft Windows (with free add-on of POSIX-like Cygwin Command Line Interface and GNU tools from Red Hat) ▪ Unix (Operating System Kernel with Unix-like Command Line Interface, Graphical User Interface and Toolchain)	▪ GNU/Linux (Linux Operating System Kernel with Unix-like GNU Command Line Interface, Graphical User Interface and Toolchain). ▪ Mac OS X (Darwin-/BSD-based Unix Operating System Kernel with GNU Command Line Interface, Toolchain and Apple Computer's Graphical User Interface and Toolchain). ▪ Microsoft Windows (with free add-on of POSIX-like Cygwin Command Line Interface and GNU tools from Red Hat) ▪ Unix (Operating System Kernel with Unix-like Command Line Interface, Graphical User Interface and Toolchain)
Terminal Device Interface Software	▪ "wxWidgets" internally uses host Operating System specific API for its Terminal Device Interface. ▪ It translates host Operating System specific events into "wxWidget" specific published API events. ▪ Events include keyboard key press / release, mouse button press / release and single / double click with mouse position at every clocked sample. ▪ Events also include window resizing.	▪ "tsWxGTUI_PyVx" internally uses Python "Curses" module specific API for its Terminal Device Interface. ▪ Python "Curses" module internally uses host Operating System specific "Curses" and/or "nCurses" API for its Terminal Device Interface. ▪ It translates "Curses" module specific events into published "wxWidget" specific API events. ▪ Events include keyboard key press / release,

Feature	"wxPython" / "wxWidgets" / "wxEmbedded"	TeamSTARS "tsWxGTUI_PyVx" Toolkit
Platform	Locally (via system console or xterm) and remotely (via vnc) accessible systems: ▪ Desktop ▪ Laptop ▪ Workstation ▪ Embedded (via KOAN's "wxEmbedded")	Locally (via system console or xterm) and remotely (via xterm) accessible systems: ▪ Desktop ▪ Laptop ▪ Workstation ▪ Embedded
		mouse button press / release and single / double / triple click with mouse position ONLY available at time of button state change. ▪ Events also include window resizing. However, event handling is currently limited to trapping the unsupported event. Though re-initializing "Curses" 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) compexities associated with re-initializing the "wxPython" emulation and the System Operator designated application program.
Programming Language	▪ "wxWidgets" is implemented in C++ ▪ "wxPython" binding/wrapper for "wxWidgets" is implemented with SWIG in Python 2.x and Python 3.x	▪ Primary baseline "tsWxGTUI_Py2x" is implemented in Python 2.x and then debugged. ▪ Secondary baseline "tsWxGTUI_Py3x" is implemented in Python 3.x as a "port" from "tsWxGTUI_Py2x via the standard Python "2to3" translation utility followed by minor manual debugging when appropriate
Terminal Interface Software	▪ "wxWidgets" internally uses Operating System or X window system specific API for Graphical User Interface.	▪ "tsWxGTUI_PyVx" internally uses Python Curses module API for Graphical-style User Interface. ▪ The Python Curses module only implements the most commonly used subset features of the host "Curses" or "nCurses" API. ▪ The "tsWxGTUI_PyVx" Toolkit emulates a typical Operating System or X window system specific API for Graphical User Interface. It establishes the appropriate relationship between a triggering event (mouse button click) and the GUI object (button, gauge, frame, dialog, panel etc.) to receive and subsequently handle the triggering event notification.

Feature	"wxPython" / "wxWidgets" / "wxEmbedded"	TeamSTARS "tsWxGTUI_PyVx" Toolkit
Platform	Locally (via system console or xterm) and remotely (via vnc) accessible systems: ▪ Desktop ▪ Laptop ▪ Workstation ▪ Embedded (via KOAN's "wxEmbedded")	Locally (via system console or xterm) and remotely (via xterm) accessible systems: ▪ Desktop ▪ Laptop ▪ Workstation ▪ Embedded
Operator Desktop Interface Software	▪ Host Operating System specific Command Line Interface is a POSIX compatible shell (such as bash) window process ▪ Host Operating System specific Graphical User Interface features support multiple independently re-sizable and repositionable Frame and Dialog processes ▪ Host Operating System specific Graphical User Interface features reflect proprietary or industry standard 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	▪ Host Operating System specific Command Line Interface is a POSIX compatible shell (such as bash) window process ▪ Host Operating System specific Graphical User Interface utilizes the existing Command Line Interface shell window process ▪ "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 window processes ▪ "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	▪ "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.	▪ "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 a single fixed sized font and at least the 68 most commonly used colors (which are internally mapped into the "Curses" standard 8-color or 16-color terminal hardware and terminal emulator software). Future "Curses" enhancements may enable support a single fixed sized font and at least the 68 most commonly used colors defined for optional 88-color and 256-color terminal hardware and terminal

Feature	"wxPython" / "wxWidgets" / "wxEmbedded"	TeamSTARS "tsWxGTUI_PyVx" Toolkit
Platform	Locally (via system console or xterm) and remotely (via vnc) accessible systems: ▪ Desktop ▪ Laptop ▪ Workstation ▪ Embedded (via KOAN's "wxEmbedded")	Locally (via system console or xterm) and remotely (via xterm) accessible systems: ▪ Desktop ▪ Laptop ▪ Workstation ▪ Embedded
		emulator software. ▪ It supports the 1-color (white, green or orange depending on the single available phosphor) that is "ON" or "OFF" (black) associated with a VT100/VT220 Terminal emulator.

2.2 High, Low and Extended API Relationships

The following tables describes the conceptual purpose, scope, capabilities, limitations and relationship between the High-level, Low-Level and Extended APIs associated with the "tsWxGTUI_PyVx" Toolkit and its third-party components.

Low Level GUI "Curses"-style API Classes and associated Class Methods:

- **Pads** - A pad is like a window, except that it is not restricted by the screen size, and is not necessarily associated with a particular part of the screen. Pads can be used when a large window is needed, and only a part of the window will be on the screen at one time. Automatic refreshes of pads (such as from scrolling or echoing of input) do not occur. The refresh() and noutrefresh() methods of a pad require 6 arguments to specify the part of the pad to be displayed and the location on the screen to be used for the display. The arguments are pminrow, pmincol, sminrow, smincol, smaxrow, smaxcol; the p arguments refer to the upper left corner of the pad region to be displayed and the s arguments define a clipping box on the screen within which the pad region is to be displayed.
- **Panels** - Panels are windows with the added feature of depth, so they can be stacked on top of each other, and only the visible portions of each window will be displayed. Panels can be added, moved up or down in the stack, and removed.
- **Windows** - a window is a rectangular area of the display with or without a border
- **Text** - A character string of one or more mono-spaced alpha-numeric, punctuation or line-draw characters.
- **Colors** - Support for terminals and terminal emulators (cygwin, linux, xterm, xterm-color, xterm-16color, xterm-88color or xterm-256color) with mouse and Red-Green-Blue color phosphors who's individual intensities can be adjusted to create a palette of terminal / terminal emulator dependent colors, ranging from 8 to 256 used to create foreground/background color pair combinations ranging from 8 x 8 to 256 x 256. However, currently the maximum number of color pairs is limited to 16 x 16.
- **Non-color** - Support for Digital Equipment Corporation VT100 and VT220 terminals and terminal emulators with/without mouse having only a single color phosphor (such as green orange or white) who's intensity can be either ON or OFF.

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher ▪ 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 ▪ login to local platform host ▪ launch local shell ▪ echo $TERM # Display default type ▪ $TERM=<xterm-family member ID> or <vt100-family member ID> ▪ echo $TERM # Display changed type ▪ launch local application ▪ terminate local application ▪ login to remote platform host via "ssh <login ID>@<Host ID>: ▪ launch remote application ▪ terminate remote application ▪ logout of remote platform host ▪ transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" ▪ terminate local shell ▪ logout of local platform host
	Local Host Operating System Process API ▪ Process Launcher ▪ Process Terminator ▪ Process Cleanup ▪ Process Event Handlers ▪ Process Input/Output Handlers	Local & Remote Host Operating System Process API ▪ Process Launcher ▪ Process Terminator ▪ Process Cleanup ▪ Process Event Handlers ▪ Process Input/Output Handlers
		Command Line Interface Low Level API (ts) ▪ tsApplication (Class and run time library component initializes and configures the application using the following keyword-value pairs and positional arguments: input provided on the command line by an operator and input provided in the parameter list of the application's invocation of the class instantiation.) ▪ tsCommandLineEnv (Class and run time library component provides platform independent configuration,

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher ▪ 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 ▪ login to local platform host ▪ launch local shell ▪ echo $TERM # Display default type ▪ $TERM=<xterm-family member ID> or <vt100-family member ID> ▪ echo $TERM # Display changed type ▪ launch local application ▪ terminate local application ▪ login to remote platform host via "ssh <login ID>@<Host ID>: ▪ launch remote application ▪ terminate remote application ▪ logout of remote platform host ▪ transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" ▪ terminate local shell ▪ logout of local platform host
		initialization, input/output supervisor and application launcher, event handler and terminator) ▪ tsCommandLineInterface (Class and methods that prompt or re-prompt the operator for input, validate that theoperator has supplied the expected number of inputs and that each is of the expected type.) ▪ tsCxGlobals (Module to establish configuration constants and macro-type functions for the Command Line Interface mode of the "tsWxGTUI_PyVx" Toolkit.) ▪ tsDoubleLinkedList (Class to establish a representation of a linked list with forward and backward pointers.) ▪ tsExceptions (Class to define and handle error exceptions. Maps run time exception types into 8-bit exit codes and prints associated diagnostic message and traceback info.) ▪ tsLogger (Class that emulates a subset of Python logging API. It defines and

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher ▪ 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 ▪ login to local platform host ▪ launch local shell ▪ echo $TERM # Display default type ▪ $TERM=<xterm-family member ID> or <vt100-family member ID> ▪ echo $TERM # Display changed type ▪ launch local application ▪ terminate local application ▪ login to remote platform host via "ssh <login ID>@<Host ID>: ▪ launch remote application ▪ terminate remote application ▪ logout of remote platform host ▪ transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" ▪ terminate local shell ▪ logout of local platform host
		handles prioritized, time and date stamped event message formatting and output to files and devices. Files are organized in a date and time stamped directory named for the launched application. Unix-type devices include syslog, stderr, stdout and stdscr (the ncurses display screen). It also supports "wxPython"-style logging of assert and check case results.) ▪ tsOperatorSettingsParser (Class to parse the command line entered by the operator of an application program. Parsing extracts the Keyword-Value pair Options and Positional Arguments that will configure and control the application during its execution.) ▪ tsPlatformRunTimeEnvironment (Class to capture current hardware, software and network information about the run time environment for the user process.) ▪ tsReportUtilities (Class defining methods used to format information:

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher ▪ 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 ▪ login to local platform host ▪ launch local shell ▪ echo $TERM # Display default type ▪ $TERM=<xterm-family member ID> or <vt100-family member ID> ▪ echo $TERM # Display changed type ▪ launch local application ▪ terminate local application ▪ login to remote platform host via "ssh <login ID>@<Host ID>: ▪ launch remote application ▪ terminate remote application ▪ logout of remote platform host ▪ transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" ▪ terminate local shell ▪ logout of local platform host
		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.) ▪ tsSysCommands (Class definition and methods for issuing shell commands to and receiving responses from the host operating system.)
Graphical User Interface Launcher ▪ 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 ▪ login to local platform host ▪ launch local shell ▪ echo $TERM # Display default type ▪ $TERM=<xterm-family member ID> or <vt100-family member ID> ▪ echo $TERM # Display changed type ▪ launch local application ▪ terminate local application ▪ login to remote platform host via "ssh <login ID>@<Host ID>: ▪ launch remote application

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher ▪ 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 ▪ login to local platform host ▪ launch local shell ▪ echo $TERM # Display default type ▪ $TERM=<xterm-family member ID> or <vt100-family member ID> ▪ echo $TERM # Display changed type ▪ launch local application ▪ terminate local application ▪ login to remote platform host via "ssh <login ID>@<Host ID>: ▪ launch remote application ▪ terminate remote application ▪ logout of remote platform host ▪ transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" ▪ terminate local shell ▪ logout of local platform host
		▪ terminate remote application ▪ logout of remote platform host ▪ transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" ▪ terminate local shell ▪ logout of local platform host
	Local Host Operating System Process API ▪ Process Launcher ▪ Process Terminator ▪ Process Cleanup ▪ Process Event Handlers ▪ Process Input/Output Handlers	Local & Remote Host Operating System Process API ▪ Process Launcher ▪ Process Terminator ▪ Process Cleanup ▪ Process Event Handlers ▪ Process Input/Output Handlers
Graphical User Interface Application Launcher API (wxWidgets & wxPython) ▪ run time library components provide platform specific configuration, initialization, input/output supervisor and application launcher	Graphical User Interface Application Launcher API (nCurses) ▪ application run time library components provide platform specific nCurses configuration, initialization, input/output supervisor and application launcher	Graphical User Interface Application Launcher API (tsWxGTUI) ▪ tsWx (Module to load all symbols that should appear within the wxPython.wx emulation namespace. Included are various classes, constants, functions and methods available for use by applications built

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher ▪ 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 ▪ login to local platform host ▪ launch local shell ▪ echo $TERM # Display default type ▪ $TERM=<xterm-family member ID> or <vt100-family member ID> ▪ echo $TERM # Display changed type ▪ launch local application ▪ terminate local application ▪ login to remote platform host via "ssh <login ID>@<Host ID>: ▪ launch remote application ▪ terminate remote application ▪ logout of remote platform host ▪ transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" ▪ terminate local shell ▪ logout of local platform host
		with components from the wxPython emulation infrastructure.) ▪ tsWxGlobals (Module to establish configuration constants and macro-type functions for the Graphical-style User Interface mode of the "tsWxGTUI_PyVx" Toolkit. Provides a centralized mechanism for modifying/restoring those configuration constants that can be interogated at runtime by those software components having a "need-to-know". The intent being to avoid subsequent searches to locate and modify or restore a constant appropriate to the current configuration. Provides a theme-based mechanism for modifying / restoring those configuration constants.) ▪ tsWxMultiFrameEnv (Class 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 uses application specified configuration keyword-value pair

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher ▪ 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 ▪ login to local platform host ▪ launch local shell ▪ echo $TERM # Display default type ▪ $TERM=<xterm-family member ID> or <vt100-family member ID> ▪ echo $TERM # Display changed type ▪ launch local application ▪ terminate local application ▪ login to remote platform host via "ssh <login ID>@<Host ID>: ▪ launch remote application ▪ terminate remote application ▪ logout of remote platform host ▪ transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" ▪ terminate local shell ▪ logout of local platform host
		options to initialize 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.)
Graphical User Interface High Level Application Launcher API (wxWidgets & wxPython) ▪ PyApp (start wxPython application) ▪ PySimpleApp (start simple wxPython application) ▪ PyOnDemandOutput (start Redirected Output window)		Graphical User Interface High Level Application Launcher API (tsWxGTUI) ▪ 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/backgtound color markup attributes)

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher ▪ 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 ▪ login to local platform host ▪ launch local shell ▪ echo $TERM # Display default type ▪ $TERM=<xterm-family member ID> or <vt100-family member ID> ▪ echo $TERM # Display changed type ▪ launch local application ▪ terminate local application ▪ login to remote platform host via "ssh <login ID>@<Host ID>: ▪ launch remote application ▪ terminate remote application ▪ logout of remote platform host ▪ transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" ▪ terminate local shell ▪ logout of local platform host
	Graphical User Interface Low Level Launcher (nCurses) ▪ start (curses.start) ▪ stop (curses.stop)	Graphical User Interface Low Level Launcher API (tsWxGTUI) ▪ start (tsWxGTUI.start) ▪ stop (tsWxGTUI.stop) ▪ tsWxGraphicalTextUserInterface (Class uses the Standard Python Curses API to initialize, manage and shutdown input, from a keyboard and mouse, and output, to a two-dimensional display screen. It identifies user terminal make, model and features. It controls terminal device startup, shutdown and exception handling. It translates "wxPython"-style terminal color, pixel and character parameters into their "Curses" counterparts. Upon startup, it briefly restores or creates, saves or loads the splash screen bitmap image as specified in the "tsWxGlobals" configuration file.)
GUI (wxWidgets & wxPython) ▪ It is a C++ library that lets developers create applications for Windows, OS X, Linux	GUI (nCurses) ▪ It is a programming library that provides an API which allows the programmer to write text mode	GUI (tsWxGTUI) ▪ It is a Python and nCurses based programming library that lets developers create wxWidgets and

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher ▪ 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 ▪ login to local platform host ▪ launch local shell ▪ echo $TERM # Display default type ▪ $TERM=<xterm-family member ID> or <vt100-family member ID> ▪ echo $TERM # Display changed type ▪ launch local application ▪ terminate local application ▪ login to remote platform host via "ssh <login ID>@<Host ID>: ▪ launch remote application ▪ terminate remote application ▪ logout of remote platform host ▪ transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" ▪ terminate local shell ▪ logout of local platform host
and UNIX on 32-bit and 64-bit architectures as well as several mobile platforms including Windows Mobile, iPhone SDK and embedded GTK+.	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.	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.
▪ It has popular language bindings for Python, Perl, Ruby and many other languages.		
▪ 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.		
▪ It's also extensive, free, open-source and mature.		

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher ▪ 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 ▪ login to local platform host ▪ launch local shell ▪ echo $TERM # Display default type ▪ $TERM=<xterm-family member ID> or <vt100-family member ID> ▪ echo $TERM # Display changed type ▪ launch local application ▪ terminate local application ▪ login to remote platform host via "ssh <login ID>@<Host ID>: ▪ launch remote application ▪ terminate remote application ▪ logout of remote platform host ▪ transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" ▪ terminate local shell ▪ logout of local platform host
▪ It supports local terminals with various keyboard, mouse and pixel mode display device configurations.	▪ It supports local and remote terminals with various keyboard, mouse and text mode display device configurations.	▪ It supports local and remote terminals with various keyboard, mouse and text mode display device configurations.
▪ It enables the operator to adjust the display position, size and appearance of the top level GUI objects.	▪ 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.	▪ It requires the operator to login to each remote computer before running an application program.
▪ It enables application programmmers to control the initial display position, size and appearance of the top level GUI objects.	▪ It enables application programmmers 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	▪ It enables application programmmers 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.

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher ▪ 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 ▪ login to local platform host ▪ launch local shell ▪ echo $TERM # Display default type ▪ $TERM=<xterm-family member ID> or <vt100-family member ID> ▪ echo $TERM # Display changed type ▪ launch local application ▪ terminate local application ▪ login to remote platform host via "ssh <login ID>@<Host ID>: ▪ launch remote application ▪ terminate remote application ▪ logout of remote platform host ▪ transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" ▪ terminate local shell ▪ logout of local platform host
	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.	
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. ▪ Linux (Fedora 17-20 & Ubuntu) 10.04-12.04 ▪ Mac OS X (Lion 10.7.4-10.9.1) ▪ Microsoft Windows (XP with SP3, 7, 8 & 8.1) with Cygwin (1.7.2) add-on	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. ▪ Python 2.6.x, 2.7.x and/or Python 3.x (provides platform independent Virtual Machine API) ▪ Python Curses Module (provides terminal independent keyboard, mouse and display API) ▪ NCurses Library (implements	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. ▪ tsWxGraphicalTextUserInterface (Class uses the Standard Python Curses API to initialize, manage and shutdown input, from a keyboard and mouse, and output, to a two-dimensional display screen. It identifies user terminal make, model and features. It controls terminal

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher ▪ 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 ▪ login to local platform host ▪ launch local shell ▪ echo $TERM # Display default type ▪ $TERM=<xterm-family member ID> or <vt100-family member ID> ▪ echo $TERM # Display changed type ▪ launch local application ▪ terminate local application ▪ login to remote platform host via "ssh <login ID>@<Host ID>: ▪ launch remote application ▪ terminate remote application ▪ logout of remote platform host ▪ transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" ▪ terminate local shell ▪ logout of local platform host
▪ Unix (SunOS 5.11)	platform specific keyboard, mouse and display API)	device startup, shutdown and exception handling. It translates "wxPython"-style terminal color, pixel and character parameters into their "Curses" counterparts. Upon startup, it briefly restores or creates, saves or loads the splash screen bitmap image as specified in the "tsWxGlobals" configuration file.)
▪	TopLevel Windows (wxWidgets & wxPython) ▪ 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.)	Top Level Windows ((tsWxGTUI implements feature(s) available in wxWidgets or wxPython library) ▪ 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, such as PasswordEntryDialog or

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher ▪ 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 ▪ login to local platform host ▪ launch local shell ▪ echo $TERM # Display default type ▪ $TERM=<xterm-family member ID> or <vt100-family member ID> ▪ echo $TERM # Display changed type ▪ launch local application ▪ terminate local application ▪ login to remote platform host via "ssh <login ID>@<Host ID>: ▪ launch remote application ▪ terminate remote application ▪ logout of remote platform host ▪ transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" ▪ terminate local shell ▪ logout of local platform host
	▪ Dialog (A GUI object, such as PasswordEntryDialog or TextEntryDialog, 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 terminated upon completion.) ▪ 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.)	TextEntryDialog, 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 terminated upon completion.) ▪ 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

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher ▪ 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 ▪ login to local platform host ▪ launch local shell ▪ echo $TERM # Display default type ▪ $TERM=<xterm-family member ID> or <vt100-family member ID> ▪ echo $TERM # Display changed type ▪ launch local application ▪ terminate local application ▪ login to remote platform host via "ssh <login ID>@<Host ID>: ▪ launch remote application ▪ terminate remote application ▪ logout of remote platform host ▪ transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" ▪ terminate local shell ▪ logout of local platform host
		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) ▪ 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 ▪ 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 ▪ login to local platform host ▪ launch local shell ▪ echo $TERM # Display default type ▪ $TERM=<xterm-family member ID> or <vt100-family member ID> ▪ echo $TERM # Display changed type ▪ launch local application ▪ terminate local application ▪ login to remote platform host via "ssh <login ID>@<Host ID>: ▪ launch remote application ▪ terminate remote application ▪ logout of remote platform host ▪ transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" ▪ terminate local shell ▪ logout of local platform host
	Low Level Windows (wxWidgets & wxPython) ▪ 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 iniating selections)	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 iniating selections)

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher ▪ 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 ▪ login to local platform host ▪ launch local shell ▪ echo $TERM # Display default type ▪ $TERM=<xterm-family member ID> or <vt100-family member ID> ▪ echo $TERM # Display changed type ▪ launch local application ▪ terminate local application ▪ login to remote platform host via "ssh <login ID>@<Host ID>: ▪ launch remote application ▪ terminate remote application ▪ logout of remote platform host ▪ transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" ▪ terminate local shell ▪ logout of local platform host
	▪ 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 ScrollBar and a scrollable text window)	▪ 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 ScrollBar and a scrollable text window)

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher ▪ 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 ▪ login to local platform host ▪ launch local shell ▪ echo $TERM # Display default type ▪ $TERM=<xterm-family member ID> or <vt100-family member ID> ▪ echo $TERM # Display changed type ▪ launch local application ▪ terminate local application ▪ login to remote platform host via "ssh <login ID>@<Host ID>: ▪ launch remote application ▪ terminate remote application ▪ logout of remote platform host ▪ transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" ▪ terminate local shell ▪ logout of local platform host
	▪ 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 action initiating selections)	▪ 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) ▪ Tool Bars (row of one or more action initiating selections)

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher ▪ 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 ▪ login to local platform host ▪ launch local shell ▪ echo $TERM # Display default type ▪ $TERM=<xterm-family member ID> or <vt100-family member ID> ▪ echo $TERM # Display changed type ▪ launch local application ▪ terminate local application ▪ login to remote platform host via "ssh <login ID>@<Host ID>: ▪ launch remote application ▪ terminate remote application ▪ logout of remote platform host ▪ transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" ▪ terminate local shell ▪ logout of local platform host
		Low Level Windows ((tsWxGTUI implements feature(s) not available in wxWidgets or wxPython library) ▪ 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 ▪ 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 ▪ login to local platform host ▪ launch local shell ▪ echo $TERM # Display default type ▪ $TERM=<xterm-family member ID> or <vt100-family member ID> ▪ echo $TERM # Display changed type ▪ launch local application ▪ terminate local application ▪ login to remote platform host via "ssh <login ID>@<Host ID>: ▪ launch remote application ▪ terminate remote application ▪ logout of remote platform host ▪ transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" ▪ terminate local shell ▪ logout of local platform host
GUI Object Styles (wxWidgets & wxPython) ▪ Overlapping ▪ Tiled ▪ Border None ▪ Border Simple ▪ Splash Screen (Bitmap Image)	GUI Object Layout Sizers (wxWidgets & wxPython) ▪ 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) ▪ 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) ▪ 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) ▪ 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	Text (tsWxGTUI) ▪ 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 ▪ 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 ▪ login to local platform host ▪ launch local shell ▪ echo $TERM # Display default type ▪ $TERM=<xterm-family member ID> or <vt100-family member ID> ▪ echo $TERM # Display changed type ▪ launch local application ▪ terminate local application ▪ login to remote platform host via "ssh <login ID>@<Host ID>: ▪ launch remote application ▪ terminate remote application ▪ logout of remote platform host ▪ transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" ▪ terminate local shell ▪ logout of local platform host
	and Underline)	
Color Palette (wxWidgets & wxPython) ▪ 68-colors ▪ 4624-foreground/background color pairs	Color Palette (nCurses) ▪ 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-	Color Palette (tsWxGTUI) ▪ 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)

High Level API	Low Level API	Extended API
Local Command Line Interface Launcher ▪ 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 ▪ login to local platform host ▪ launch local shell ▪ echo $TERM # Display default type ▪ $TERM=<xterm-family member ID> or <vt100-family member ID> ▪ echo $TERM # Display changed type ▪ launch local application ▪ terminate local application ▪ login to remote platform host via "ssh <login ID>@<Host ID>: ▪ launch remote application ▪ terminate remote application ▪ logout of remote platform host ▪ transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" ▪ terminate local shell ▪ logout of local platform host
	character build of nCurses and Python Curses modules) Non-Color Palette (nCurses) ▪ 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)	▪ 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)