BSSE Autopilot

The Platform for Automated Testing and Operation of GUIs

BSSE AutoPilot (AP) supports automated operation and testing of Graphical User Interfaces (GUI). By AP a user can initiate mouse and keyboard interactions from predefined sequences represented by scripts. Scripts can be generated by recording of user interactions, by manually writing a script based on an interpreting language (APSL) close to C/C++ syntax, and by automated code generation from data included in data tables like spreadsheets or from user-provided templates. Execution of GUI actions is either time-triggered or event-triggered. The time scale can be changed so that slow-motion is possible during testing, while for real operation a higher speed can be turned on. AP provides meaningful, randomly generated mouse and keyboard inputs.

AutoPilot takes any GUI as black-box and does not require source code. AP collects data from the interface between a GUI and the operating system (OS) and complements this information by what becomes visible on the screen. Even if not supported by the OS, AP forms logical groups of graphical elements, e.g. it correlates text aside of a dialog input box with the box itself to an entity which can be accessed by the assigned text. AP just evaluates the available information and is then immediately ready to accept commands from scripts. Such information can also be retrieved from its database (APDB) which can steadily grow. APDBs can be exchanged between different users and sites, so that not every user needs to collect data on GUIs. From the database browser a user can interactively correlate object properties with an object's position on the screen. Vice versa, he can refer to a database entry by the object's position on the screen.

AutoPilot provides the information needed to operate a GUI in a generic manner: loops over all known GUIs, all menus and windows of a GUI, all objects of a window are possible. Window objects may be selected by the object type, by their distance from an object, by direction or by refering to logical entities associated with an object such as rows and columns of a checkbox matrix.

Image snapshots can be taken from any rectangle associated with a window (including the desktop) and saved to and retrieved from file in PIC-format. From two rectangles a differential pattern can be derived and be displayed, saved and retrieved. For each image color profiles are generated.

AutoPilot will support network functions and remote script execution between Mac OS and Mac OS - MS-Windows platforms.

Finally, AutoPilot provides means for verification of succesful completion of actions, or for analysis of system status, tracks the hits on window objects by mouse and keyboard and provides summary figures on counts and coverage figures on the level of window objects, windows, GUIs and top system level.

The following list summarises the most essential capabilities of AutoPilot:

- **APSL: an interpreting C/C++-like language** with script compilation and strong typing, a library of more than 400 functions to interact with GUIs, interface to OS and C functions, support of remote operations
- **script generation:** manual generation and automated generation from spreadsheets and data tables, by recording of user actions, or by the AutoPilot Script Wizzard
- **time- and event-triggered execution** of GUI operations, playback of recorded actions, execution of script hierarchies, parallel script execution (asynchronous, synchronised), sequential execution, synchronisation by semaphores, launch / quit of applications,
- **automated correlation** of window objects, e.g. icons and input fields with text, identification of (anonymous) matrix elements like check boxes by row and column names, access of anonymous objects by relative movement, n-th object closest to a certain object regarding a certain direction etc, identification of the correct coordinates of an object within a window, autoscrolling of invisible window objects into the visible part of a window (these features are of high importance for Mac OS which does not provide them itself)
- **database browser**: browse GUIs, windows, controls, dialogs, menus, show properties of objects, edit objects, move to a control or dialog field via a database entry, identify the database entry of a control or dialog field

- **data management**: support of global and local environment variables, script command parameters, include-, exec- and other user-defined paths
- **file handling, string handling and pattern matching**
- **archiving** of data, logging of information
- **image processing**, snapshots of window contents, saving/retrieval in PIC-format, differential patterns of a pair of images

  This feature alos allows to identify the status of e.g. a button by comparing the object's color spectrum against a reference.
- **configuration management** by AP's GUI, from within scripts and from AP's database files, exchange of database and configuration information
- **test and verification management**: tracking of mouse and keyboard hits, verification of executed actions and analysis of system status
- **identification of anonymous objects:** these are GUIs or windows of which the name is not known in advance, but for which its name need to be identified for further operation. AutoPilot allows to identify the actual names of such objects by comparison of the system status before and after their launch or creation
- **support of alias** for window names and names of window objects
  In case a window's or a button's title changes during script execution or is undefined at all, a user-defined alias still allows to access the object from within a script on a logical level
- **generation of "meaningful" random patterns** from user-defined data, e.g. to provide a large number of inputs for dialog fields. In this context he definition of "meaningful" is: the pattern is composed of sub-patterns as provided by the user, e.g. an address is randomly composed of sub-patterns for first name, family name, street, city and so on.
- **automated script generation** from user-provided templates and user-defined data types:
  Scripts which convert between enumeration types and integer and string are autoamtically generated for any enumeration type defined by a user. Also, scripts can be generated automatically for any set of user-defined operations for a given set of user-defined data types.
- **Script Wizard**
  The Script Wizard of AutoPilot inserts structural blocks like "if", "for", "while", main body, file header, calls of APSL library functions and the needed data declarations into a script file. A function may either directly chosen from a menu or by selction of a GUI item like window and control in the database viewer. When selecting an item like a control, the Script Wizard (1) displays a list of functions (methods) related to this item, and (2) automatically passes the actual values of GUI, window and control as parameters to the selected function.
- **event tracing and conditional wait on events** for events like launch and quit of GUIs, opening and closing of windows, initiating of actions when the event occurs, occurence of a certain text pattern. Tracing of events makes visible all background operations on GUIs and windows and gives debugging support to understand what is going on during script execution
- **unique platform for GUI testing** on Mac OS 9 / Mac OS X and MS-Windows
- **coherent network-wide GUI operation** by scripts and remote script execution
- **support of operations on window hierarchies**
  windows which include overlays of child windows (yielding a window hierarchy) can be operated by a pattern composed out of window names and buttons/actions which create the window. To press a button of a child window on a lower hierarchy level a user only needs to pass the pattern identifying the button (the "path" to the button) and AP will care about the navigation through the window hierarchy.

Platforms: Mac OS 9 (now)**, Mac OS X** (now)**, MS-Windows** *(on request)*

MAC OS GUI TYPES VS. AUTOPILOT SUPPORT (AUTOPILOT V 2.3.00)

Functionality (Part 1 of 3)	Carbon	Cocoa	Java
General Functionality			
Mouse and keyboard operations	supported	supported	supported
Recording/capture	supported	supported	supported
Playback	supported	supported	supported
Script execution	supported	supported	supported
Concurrent script execution	supported	supported	supported
Script include files	supported	supported	supported
Script functions	supported	supported	supported
Pattern matching / string handling	supported	supported	supported
File handling	supported	supported	supported
Script environment variables	supported	supported	supported
Script include and exec paths	supported	supported	supported
Script command parameters	supported	supported	supported
Configuration parameters	supported	supported	supported
Log files (events, errors, warnings)	supported	supported	supported
Test coverage	supported	supported	supported
Test case generation	supported	supported	supported
Mathematical script library	supported	supported	supported
Attach user library	supported	supported	supported
Pixel and image processing	supported	supported	supported
Archiving	supported	supported	supported
Test coverage	supported	supported	supported
Database Viewer	supported	supported	supported
Script Wizard	supported	supported	supported
Automatic code generation	supported	supported	supported
Recording of user-defined messages	supported	supported	supported
Recording of events, user actions, scripts actions, functions calls)	supported	supported	supported
Remote operation	in part	in part	in part
Process/GUI			
automatic identification	supported	supported	supported
pattern matching on GUI name			
menu identification	supported	supported	supported
logical access of processes	supported	supported	supported
number of windows	supported	supported	supported
launch	from script, by Dock, by menu	from script, by Dock, by menu	from script, by Dock, by menu
quit	from script	from script	from script
verification launch/quit	supported	supported	supported
Menu			
automatic identification of menus	supported	supported	supported
logical access of menus	supported in various kinds	supported in various kinds	supported in various kinds

Functionality (Part 2 of 3)	Carbon	Cocoa	Java
Window			
automatic identification	supported	supported	supported
pattern matching on window name	supported	supported	supported
identification of window owner	supported	supported	supported
window hierarchy (sub-windows)	supported	supported	supported
dragging	supported	supported	supported
change of size	supported	supported	supported
close box	supported	supported	supported
drag area	supported	supported	supported
grow box	supported	supported	supported
zoom box	supported	supported	supported
collapse box	supported	supported	supported
access by title	supported	supported	supported
access by alias	supported	supported	supported
access by numerical id	supported	supported	supported
boundaries, relative	supported	supported	supported
boundaries, absolute	supported	supported	supported
test coverage	supported	supported	supported
window objects (controls, dialogs)	supported	supported	user-defined, by position, as recorded
text as window objects	supported	supported	supported
automatic identification of window objects	supported	supported	not supported
number of controls	supported	supported	supported
number of dialog fields	supported	supported	supported
open window	from script by menu and control	from script by menu and control	from script by menu and control
close window	from script by menu, control and close box	from script by menu, control and close box	from script by menu, control and close box
verification open/close	supported	supported	supported
save image of a window (section)	supported	supported	supported
comparison of images	supported	supported	supported
status zoom	supported	not supported	not supported
status collapse	supported	not supported	not supported
status visibility	supported	not supported	not supported

Functionality (Part 3 of 3)	Carbon	Cocoa	Java
Controls			
automatic identification	supported	supported	not supported
text as control	supported	supported	not applicable
access by title	supported	supported	user-defined controls only
access by alias	supported	supported	user-defined controls only
access by numerical id	supported	supported	user-defined controls only
access by position	supported	supported	supported
boundaries, window relative	supported	supported	user-defined controls only
test coverage	supported	supported	user-defined controls only
save image of a control (section)	supported	supported	user-defined controls only
comparison of images	supported	supported	user-defined controls only
Dialog Fields			
automatic identification	supported	supported	not supported
access by title	supported	supported	user-defined dialogs only
access by alias	supported	supported	user-defined dialogs only
access by numerical id	supported	supported	user-defined dialogs only
access by position	supported	supported	supported
boundaries, window relative	supported	supported	user-defined dialogs only
test coverage	supported	supported	user-defined dialogs only
save image of a dialog field (section)	supported	supported	user-defined dialogs only
comparison of images	supported	supported	user-defined dialogs only
provision of inputs	supported	supported	supported
recording of inputs	supported	supported, AP copy	as for Cocoa, user-defined dialogs only
retrieval of inputs	supported	supported AP copy	as for Cocoa, user-defined dialogs only