COMARCH

Implementation of Standard Accessibility APIs for Tizen

Dariusz Filipski

Krzysztof Włodarczyk

TIZEN™ DEVELOPER CONFERENCE
MAY 7–9, 2012

COMARCH

Agenda

- Accessibility - what is it and why is it important?
- Main accessibility standards
- Current status of accessibility in EFL
- AT-SPI – infrastructure, ATK interfaces, AT-SPI vs. AT-SPI2
- Porting AT-SPI2, ATK interface and the bridge
- EAIL – Enlightenment Accessibility Implementation Library – architecture, implementation details, a11y support for widgets,
- EAIL initialisation
- Issues we faced so far and proposed solutions and workarounds
- Accessibility and WebKit/EFL
- Live demo
- Example use case
- Q&A

Definition of accessibility

Accessibility is a general term used to describe the degree to which a product, device, service, or environment is available to as many people as possible. Accessibility can be viewed as the "ability to access" and benefit from some system or entity. Accessibility is often used to focus on people with disabilities or special needs and their right of access to entities [1]

Computer accessibility (in terms of computer hardware and software), often abbreviated to a11y, refers to ability of a computer system to be accessible for all people, regardless of disability or severity of impairment. Software, hardware or a combination of software and hardware that enable disabled or impaired people to use a computer are known as **Assistive Technologies**. [2]

[1] http://en.wikipedia.org/wiki/Accessibility
[2] http://en.wikipedia.org/wiki/Computer_accessibility

Introduction

Comarch involved in work on a11y since 2006 to provide a unified solution for the disabled and a handy interface for automated UI testing

- Developed accessibility based solutions for test automation (including open-sourced TADEK). Many of these solutions use client AT-SPI libraries for Python: PySpi, PyAtSpi for AT-SPI and PyAtSpi2 for AT-SPI2
- Implemented ATK interfaces for GTK+ based libraries, such as SAIL, RAIL
- Contributed to the standalone implementation of GAIL
- Implemented QAccessibleInterface set of interfaces for Qt/QML, QtWebKit, and QtWebKit2
- Modified Qt AT-SPI bridge to provide lazy internal caching in the bridge
- Experienced on both AT-SPI (CORBA-based) and AT-SPI2 (DBus-based) implementations
- Cooperated with open source community - implementation, testing, providing feedback, and patches

Why accessibility is important? 1/3

- The disabled
  - 30 million people whose ability to use computers may be compromised by inaccessible design in the US alone. Globally, ~8% people using the Internet have some sort of disability [1]
    - Visual impairments – from low-vision to complete blindness
    - Movement impairments - poor muscle control or weaknesses
    - Hearing impairments - from problems with some sounds to deafness
    - Cognitive and language impairments - from dyslexia to difficulties remembering things, solving problems or comprehending and using spoken or written language
    - Seizure disorders - certain light or sound patterns can cause epileptic seizures in susceptible users
- Legal obligations
  - Legal obligations vary from country to country but generally it is necessary [2], for example:
    - a human or civil right to Communications Technology (ICT)
    - any ICT purchased by government must be accessible
    - any ICT sold in a given market must be accessible
    - etc.
  - Section 508 of The Rehabilitation Act Amendments in the US [3] –
    if you want to sell for US Public Sector (Government, Agencies, Universities, etc.)
    your product must to be accessible

[1] http://developer.gnome.org/accessibility-devel-guide/3.0/id404485.html.en
[2] http://www.w3.org/WAI/Policy/
[3] http://www.access-board.gov/sec508/guide/act.htm

Why accessibility is important? 2/3

- Standard software and hardware
  - Screen readers
    - Orca – http://live.gnome.org/Orca
    - LSR – Linux Screen Reader - http://live.gnome.org/LSR
  - Screen magnifiers
    - Orca
    - GNOME Magnifier (gnome-mag)
  - Refreshable Braille displays [1] [2], BrlTTY
  - On-screen keyboards
  - Voice control and speech recognition
  - OCR
  - Alternative HID like Wii remote, head-mouse, eye tracker
- Alternative use cases
  - Read emails with voice synthesizer when stuck in a traffic jam
  - Live translation of non-localized software, pictures from the camera
  - IVI control
  - ...

[1] http://en.wikipedia.org/wiki/Refreshable_Braille_display , photo Ralf Roletschek Marcela
[2] http://www.youtube.com/watch?v=Gd10syL5RLY , Victor Tsaran, Yahoo! Accessibility Lab

Why accessibility is important? 3/3

- Testability
  - Dogtail –https://fedorahosted.org/dogtail/
  - LDTP – Linux Desktop Testing Project - http://ldtp.freedesktop.org/wiki/
  - TADEK – Test Automation in a Distributed Environment - http://tadek.comarch.com/
- A11y in framework gives virtually "free" support for AT in applications
- GTK has it, Qt has it, why not EFL?

COMARCH

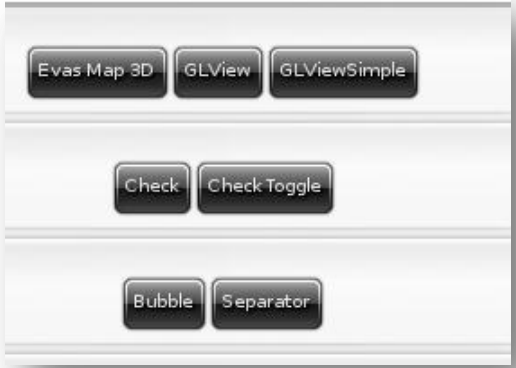
Main accessibility standards

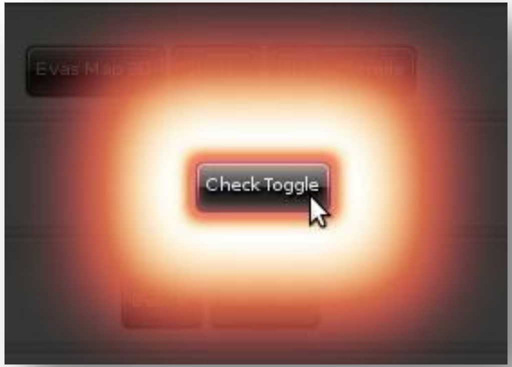
- AT-SPI
    - Toolkit-neutral, developed by the GNOME project, considered as a standard on Linux and Unix [1]
- Microsoft Active Accessibility (MSAA)
    - Introduced in 1997 for Windows 95, based on the Component Object Model (COM)
- IAccessible2
    - Initially developed by IBM, extending MSAA when creating ODF based office suite
    - Under The Linux Foundation, Open A11y Workgroup, harmonized with AT-SPI [2]
- Microsoft UI Automation
    - Similar to MSAA but with much richer object
- Apple Accessibility API
    - Accessibility objects, their attributes and hierarchy, actions
- Java Accessibility API
    - Java Access Bridge

[1] http://accessibility.linuxfoundation.org/a11yweb/soi.html
[2] http://www.linuxfoundation.org/collaborate/workgroups/accessibility/iaccessible2

Current status of a11y in EFL

- Elementary has some built-in accessibility features:
  - *ELM_ACCESS_MODE=1*
  - Glowing outline of widgets
  - *ELM_MODULES="access_output>access/api"*
  - Type and text are read by a voice synthesiser
- None of accessibility standards are supported
- There is little to configure
- Flat maturity curve
- Possible advantages of adapting a standard:
  - Avoid reinventing the wheel
  - One accessibility standard for native applications and HTML5 applications
  - Numerous already existing assistive technology applications available
  - Open way to implement interesting use cases





COMARCH

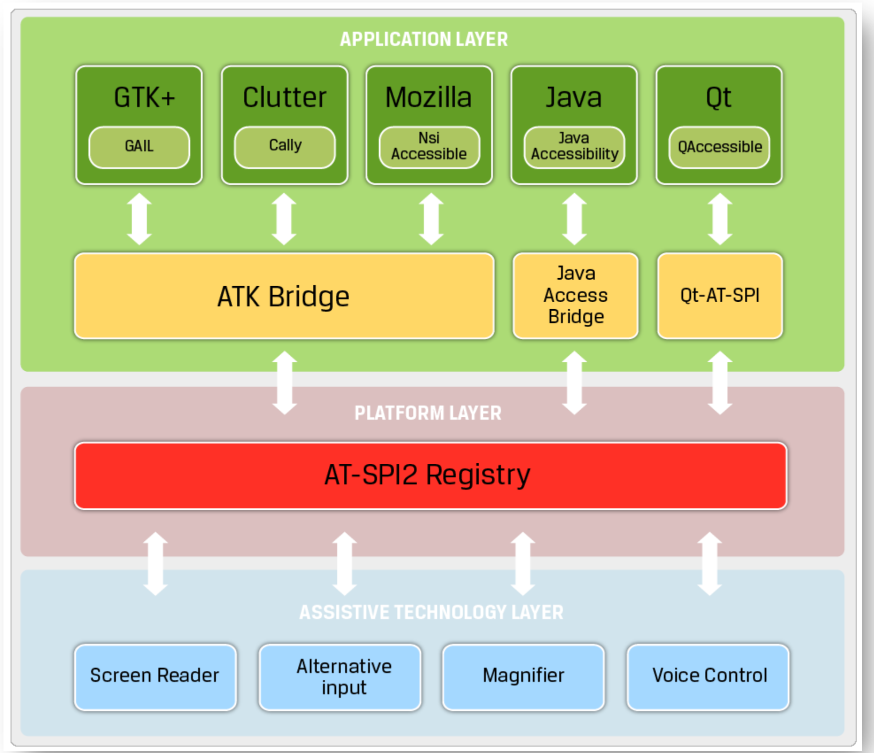
More on AT-SPI

- Assistive Technology Service Provider Interface - a standardized interface between assistive technologies and user's desktop and applications
- Originates from the GNOME desktop environment
- It is toolkit-agnostic
- Currently supported by: GTK+, Java/Swing, the Mozilla suite, StarOffice/OpenOffice.org and Qt 4 [1]
- Benefits on using AT-SPI
  - Compatibility with existing and well-recognized solutions
  - Interoperability between assistive technologies

[1] http://www.linuxfoundation.org/collaborate/workgroups/accessibility/atk/at-spi

More on AT-SPI – the architecture

- Application layer: user applications (servers)
- Platform layer: AT-SPI registry – central point of the architecture
- Assistive Technology layer: AT software (clients)
- ATK – toolkit independent interfaces for describing and interaction with application widgets
- ATK implementations for specific toolkits
  - With ATK-Bridge – the most simple way – GAIL, Cally, nsiAccessible
  - Without ATK-Bridge – Java Accessibility + Java Accessibility Framework, Qaccessible + Qt-ATK bridge



More on AT-SPI – ATK interfaces 1/2

The role of ATK interfaces is to provide an accessible object for each widget in an application

- *AtkObject* – for all widget types
  - parent-child relations, name, role, description, states, relations
- *AtkComponent* – for widgets that occupy a physical area on the screen
  - size, position, grabbing focus
- *AtkText* – for widgets containing text
  - text, text at/before/after offset or X/Y position, selections, caret offset
  - labels, text edits, buttons

More on AT-SPI – ATK interfaces 2/2

- *AtkAction* – for widgets having actions like click, press, release, jump, and activate
  - actions count, action name, description, key binding, execute action
  - buttons, check boxes, text edits
- *AtkValue* – for widgets representing a numeric value from a bounded range of values
  - value, maximum, minimum, increment
  - spin boxes, sliders, scroll bars
- *AtkEditableText*, *AtkImage*, *AtkHypertext*, *AtkHyperlink*, *AtkTable*, *AtkDocument*

AT-SPI vs. AT-SPI2

- AT-SPI
  - Uses CORBA, which is deprecated and being replaced by newer technologies like D-Bus
  - Not developed any more
- AT-SPI2
  - Uses D-Bus for inter-process communication
  - The registry holds widget cache to compensate slower IPC performance
  - Stable version, under continuous development
  - API compatible with previous version

Porting AT-SPI2

- AT-SPI2 binaries:
  - *libatspi2.0* – library with definition of AT-SPI interface
  - *at-spi2-core* – D-bus services:
    - *at-spi-bus-launcher* (*org.a11y.Bus*) – maintains the lifecycle of a11y bus and provides a method for retrieving its address, the bus is used to talk with applications in AT-SPI protocol
    - *at-spi2-registryd* (*org.a11y.atspi.Registry*) – keeps track of accessible applications, forwards X events over a11y bus, the registry bus is used by assistive technologies to discover applications and send queries
- AT-SPI2 development files:
  - *at-spi2.0-dev* – sources required to include when implementing an AT client application
- Tasks:
  - Provide dependencies:
    - *libatspi2.0*: *libc6, libdbus, libglib2.0, libx11*
    - *libatspi2.0-dev*: *libglib2.0-dev, libdbus-1-dev, libdbus-glib-1-dev, dbus, libxtst-dev*
    - *at-spi2-core*: *libc6, libdbus, libglib2.0, libx11, libxtst6*
  - Prepare source and RPM packages with latest stable versions

Porting ATK interface

- ATK interface shared library:
  - *libatk1.0* – library with runtime part of ATK
  - Needed to load libraries containing ATK implementations or to run applications that provide their own implementation of ATK interfaces
- ATK interface development files
  - *libatk1.0-dev* – sources required to include when implementing ATK interfaces
- Tasks:
  - Provide dependencies:
    - *libatk1.0*: *libc6, libglib2.0, libc6, libdbus-1, libglib2.0, libx11, libxtst6*
  - Prepare source and RPM packages with latest stable versions

Porting ATK bridge

- ATK-bridge - also known as atk-adaptor
  - Handles D-bus communication between AT-SPI2 registry and ATK
  - Not necessarily required, but it makes it easier to create a new ATK implementation
- Tasks:
  - Provide dependencies:
    - *atk-adaptor*: *libc6, libdbus-1, libglib2.0,* ***dconf-gsettings-backend*** (or another configuration system interface)
  - Prepare source and RPM packages with latest stable versions

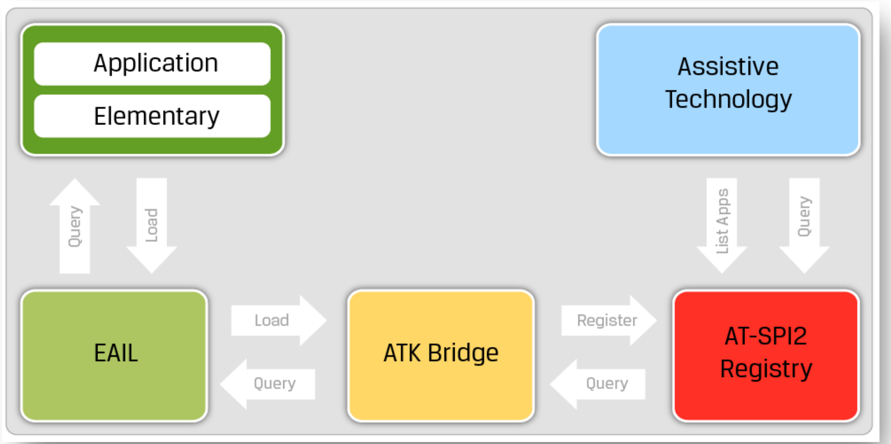
COMARCH

ATK interface for Elementary

EAIL – Enlightenment Accessibility Implementation Library

(inspired by GAIL – Gnome Accessiblity Implementation Library)

- Loads ATK bridge
- Integrates the glib event loop with the ecore main loop
- Implements ATK:
  - Core functionality – provides an accessible application object, toolkit information, and registering events
  - Factory function that creates an accessible widget object based on a given Elementary widget
  - Basic accessible widget type representing a widget with *AtkObject* and *AtkComponent* interfaces
  - Specialized accessible widget types for specific widgets of an application
- Provides a function for initializing the library



ATK interface for Elementary – AtkUtilClass 1/2

- Implement utility functions related to toolkit and event support in *AtkUtilClass*:

```
static void
atk_util_install(void)
{
    AtkUtilClass *uclass;

    uclass = ATK_UTIL_CLASS(g_type_class_ref(ATK_TYPE_UTIL));
    uclass->get_toolkit_name = eail_get_toolkit_name;
    uclass->get_toolkit_version = eail_get_toolkit_version;
    uclass->get_root = eail_get_root;
    /*
     * TODO: Define functions:
     */
    uclass->add_global_event_listener = NULL;
    uclass->remove_global_event_listener = NULL;
    uclass->add_key_event_listener = NULL;
    uclass->remove_key_event_listener = NULL;
}
```

COMARCH

ATK interface for Elementary – AtkUtilClass 2/2

- Get toolkit name
- Get toolkit version
- Get application object

```
static const gchar * eail_get_toolkit_name(void)
{
    return "elementary";
}


static const gchar * eail_get_toolkit_version(void)
{
    return g_strdup_printf("%i.%i.%i", elm_version->major,
                                       elm_version->minor,
                                       elm_version->micro);

}

static AtkObject * eail_get_root(void)
{
    static AtkObject *root = NULL;

    if (!root) {
        root = g_object_new(EAIL_TYPE_APP, NULL);
        atk_object_initialize(root, NULL);
    }

    return root;
}
```

ATK interface for Elementary – accessible widget factory

- Takes *Evas_Object* and returns *AtkObject*
- Obtains type name using *elm_object_widget_type_get()* function from Elementary API
- Creates and initializes an object of one of the accessible object types based on widget type name

```
AtkObject * eail_factory_get_accessible(Evas_Object *widget)
{
    const char *type = NULL;
    AtkObject *accessible = NULL;

    type = elm_object_widget_type_get(widget);
    if (!strcmp(type, "win")) {
        accessible = g_object_new(EAIL_TYPE_WINDOW, NULL);
    } else if (!strcmp(type, "box")) {
        accessible = g_object_new(EAIL_TYPE_BOX, NULL);

                              ...

    } else {
        accessible = g_object_new(EAIL_TYPE_WIDGET, NULL);
    }

    if (accessible) {
        atk_object_initialize(accessible, widget);
    }

    return accessible;
}
```

ATK interface for Elementary – widget base

EailWidget – base of all accessible widget types, implements AtkObject and AtkComponent

- *get_n_children, ref_child* – for widgets containing children or default content
- *get_name – evas_object_name_get()*
- *get_parent – elm_object_parent_widget_get()*
- *ref_state_set*

ATK_STATE_SENSITIVE, ATK_STATE_ENABLED	*elm_object_disabled_get()*
ATK_STATE_VISIBLE	*evas_object_visible_get()*
ATK_STATE_SHOWING	*evas_object_geometry_get(), evas_output_viewport_get()*
ATK_STATE_FOCUSABLE	*elm_object_focus_allow_get()*
ATK_STATE_FOCUSED	*elm_object_focus_get()*

- *get_attributes* – additional key-value information

type	*elm_object_widget_type_get()*

ATK interface for Elementary – specific widgets

- Accessible widgets are created using the factory function
- They derive from *EailWidget*
- They implement one or more ATK interfaces, for example
  - *set_text_contents* method of *AtkEditableText* uses *elm_object_text_set()*
  - *set_current_value* method of *AtkValue* uses *elm_slider_value_set()*
- Missing *ref_child* and *get_n_children* of *AtkObject* use Elementary API functions:
  - *elm_box_children_get()* for Box widget or similar for other explicit containers
  - *elm_object_part_content_get()* for widgets containing default content
  - Low level Evas functions as a workaround for accessing sub-elements of Win objects
- More states are added in *ref_state_set*
- More attributes are added in *get_attributes*, for example
  - Bg widgets "display mode" is set with "center", "scale", "stretch" or "tile" based on enumeration value returned from *elm_bg_option_get()*

COMARCH

ATK interface for Elementary - initialisation of a11y module 1/2

Since EAIL is not a part of Elementary, it should be loaded only when needed.
There are 3 ways to activate the library:

a) By implementing it as an alternative to the *access_output* module of Elementary

- Pros:
  - No modifications in Elementary
- Cons:
  - Module stays inactive until the cursor is moved over an interactive widget
  - Appearance and behavior of widgets is altered by the built-in accessibility features of Elementary

b) By loading the library from the actual application code

- Pros:
  - No modifications in Elementary
- Cons:
  - Application sources have to be modified and recompiled

ATK interface for Elementary - initialisation of a11y module 2/2

c) By creating a gate inside Elementary – the preferred way

- Patch for Elementary sources will contain loading and initialization of EAIL library inside the *elm_run()* function
- EAIL will be loaded only if *ELM_ACCESSIBILITY* environment variable is set
- Pros:
    - No modifications in application sources
    - Accessibility can be used on demand
- Cons:
    - Elementary sources have to slightly modified

ATK interface for Elementary - issues

- No function for retrieving the top level widget list, low level Ecore_Evas API have to be used:
  - *ecore_evas_ecore_evas_list_get()*
  - *core_evas_object_associate_get()*
- No common API for accessing a widget's children
- Some widgets do not have a function for retrieving the children list, for example the Win, Evas API functions provide a workaround:
  - *evas_object_geometry_get()*
  - *evas_objects_in_rectagle_get()*
- Some events can be triggered only from the UI, for example, the "activate" event for the Entry widget

A11y for WebKit/EFL

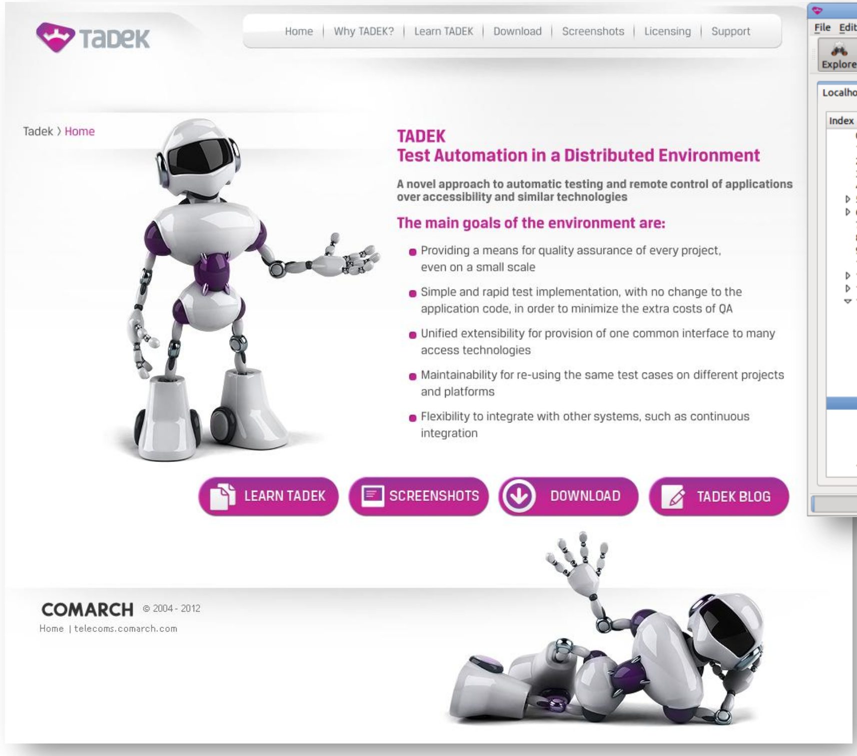
- Why WebKit?
  - HTML5 applications are rendered using the EFL port of the WebKit engine
- ATK interfaces are already implemented in WebKitGtk+
  - http://webkitgtk.org
  - *WebCore/Accessibility/Gtk* in WebKitGtk+ sources
- After AT-SPI2 is ported to Tizen, it should be easy to port the ATK implementation from WebKitGtk+ to EFL WebKit

LIVE DEMO

Use case example – TADEK

TADEK – Test Automation in a Distributed Environment

http://tadek.comarch.com/

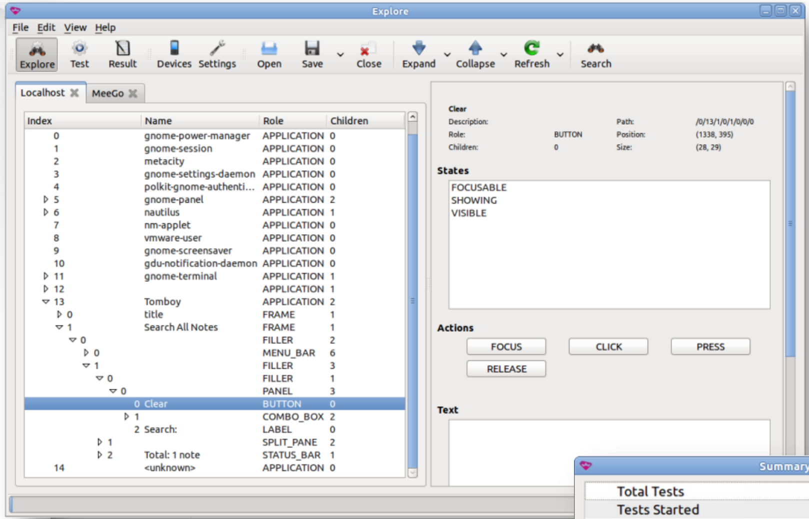

TADEK
Home | Why TADEK? | Learn TADEK | Download | Screenshots | Licensing | Support
Tadek › Home
LEARN TADEK
SCREENSHOTS
DOWNLOAD
TADEK BLOG
COMARCH © 2004 - 2012
Home | telecoms.comarch.com

TADEK
Test Automation in a Distributed Environment

A novel approach to automatic testing and remote control of applications over accessibility and similar technologies

The main goals of the environment are:

- Providing a means for quality assurance of every project, even on a small scale
- Simple and rapid test implementation, with no change to the application code, in order to minimize the extra costs of QA
- Unified extensibility for provision of one common interface to many access technologies
- Maintainability for re-using the same test cases on different projects and platforms
- Flexibility to integrate with other systems, such as continuous integration


Explore
File Edit View Help
Explore Test Result Devices Settings Open Save Close Expand Collapse Refresh Search
Localhost
MeeGo
Actions
FOCUS
CLICK
PRESS
RELEASE
Text

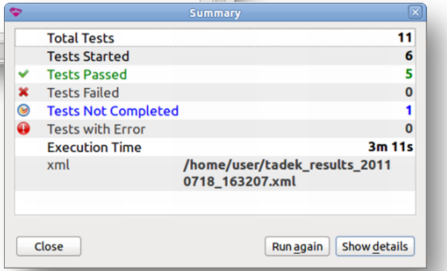
Index	Name	Role	Children
0	gnome-power-manager	APPLICATION	0
1	gnome-session	APPLICATION	0
2	metacity	APPLICATION	0
3	gnome-settings-daemon	APPLICATION	0
4	polkit-gnome-authenti...	APPLICATION	0
▷ 5	gnome-panel	APPLICATION	2
▷ 6	nautilus	APPLICATION	1
7	nm-applet	APPLICATION	0
8	vmware-user	APPLICATION	0
9	gnome-screensaver	APPLICATION	0
10	gdu-notification-daemon	APPLICATION	0
▷ 11	gnome-terminal	APPLICATION	1
▷ 12		APPLICATION	1
▽ 13	Tomboy	APPLICATION	2
▷ 0	title	FRAME	1
▽ 1	Search All Notes	FRAME	1
▽ 0		FILLER	2
▷ 0		MENU_BAR	6
▽ 1		FILLER	3
▽ 0		FILLER	1
▽ 0		PANEL	3
0	Clear	BUTTON	0
▷ 1		COMBO_BOX	2
2	Search:	LABEL	0
▷ 1		SPLIT_PANE	2
▷ 2	Total: 1 note	STATUS_BAR	1
14	<unknown>	APPLICATION	0

Clear

Description:		Path:	/0/13/1/0/1/0/0/0
Role:	BUTTON	Position:	(1338, 395)
Children:	0	Size:	(28, 29)

States

FOCUSABLE
SHOWING
VISIBLE



	Total Tests	11
	Tests Started	6
✓	Tests Passed	5
✗	Tests Failed	0
●	Tests Not Completed	1
●	Tests with Error	0
	Execution Time	3m 11s
	xml	/home/user/tadek_results_20110718_163207.xml

Q & A

COMARCH

Thank You!

Contact Details:

Dariusz Filipski
R&D Director, Open Source - Mobile Solutions & Services
✉ dariusz.filipski@comarch.com

Krzysztof Włodarczyk
Developer, Open Source - Mobile Solutions & Services
✉ krzysztof.j.wlodarczyk@comarch.com