OPEN SOURCE
MONITORING
CONFERENCE

22.-24. OCTOBER 2013 | NUREMBERG

WÜRTHPHOENIX

End User Experience Monitoring with

alexa

licensed under GNU General Public License version 3

by Georg Kostner

…sponsored by Würth Phoenix

- IT and Consulting Company of the Würth-Group
- Headquarter in Italy, European-wide presence, more than 100 employees
- International experience in Business Software and IT Management
- Core competencies in trading processes, wholesale distribution, logistics and System Monitoring
- ITIL certified, Nagios Solution Provider, Microsoft Gold Certified Partner

Facts & figures

- More than 600 customers worldwide
- Over 7.000 ERP and CRM users
- 25.000 monitored hosts
- 4 offices in 3 countries
- HQ in Italy
- Core offers in Business Software and IT System Management

… Würth Phoenix belongs to the Würth group

- The Würth Group is world market leader in its core business, the trade in assembly and fastening material
- It currently consists of over 410 companies in 84 countries and has more than 65,000 employees on its payroll. Over 30,000 of these are permanently employed sales representatives.
- In the first half of the business year 2011, the Würth Group generated total sales of EUR 4.78 billion.
- The headquarter of the Würth Group, Adolf Würth GmbH & Co. KG, was founded by Adolf Würth in 1945 in Künzelsau in Baden-Württemberg, Germany.

Facts & figures

- More than 65,000 employees worldwide
- 30, 000 sales representatives
- More than 100.000 products

WÜRTH GROUP

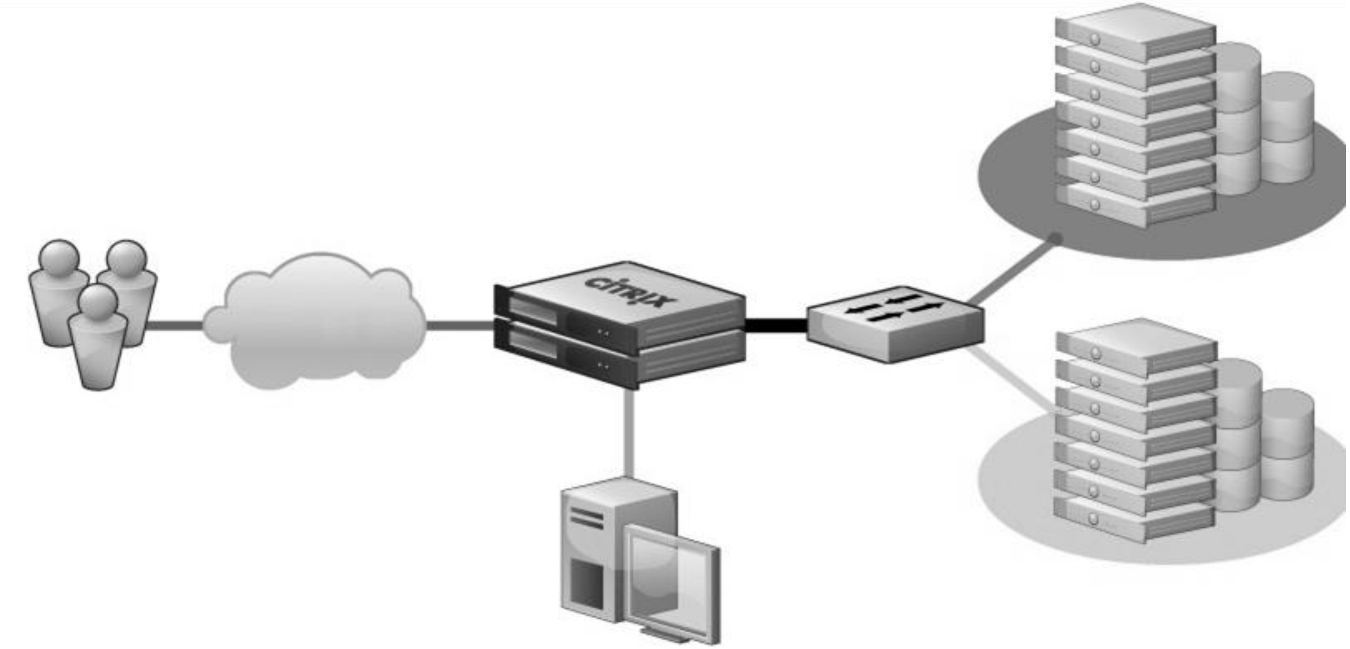
OPEN SOURCE MONITORING CONFERENCE

About the Al'exa founder

- In 2012 Alan Pipitone created a company with the primary purpose of providing consultancy applied to the world of robotics and automation
- By combining the experience gained in the world of automation with the experience gained in the information technology area, intelligent monitoring systems have been created
- Alan Pipitone started to produce custom software to simulate human behavior necessary to monitor specific applications or to interact with them
- In 2012 the Open Source project of Al'exa was born in collaboration with Diesel S.p.A. and Würth Phoenix S.r.l.

ALAN PIPITONE

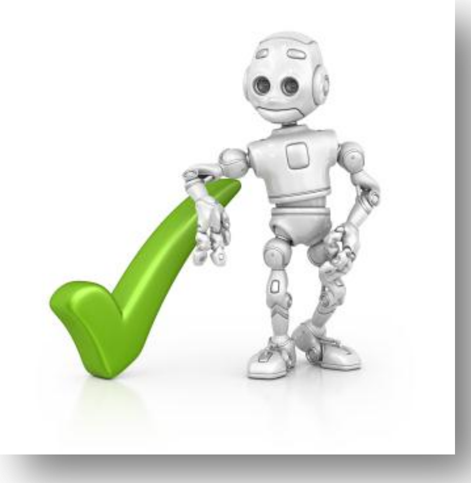
The initial requirement



How is it possible to test a Citrix farm from the users' perspective

The idea to develop Al'exa was born…

- Automate the applications through the simulation of the user interactions on a desktop
- Verify the availability and reliability of user applications and gather performance data
- Get stable test cases as much as possible against application changes
- Create an Open Source solution

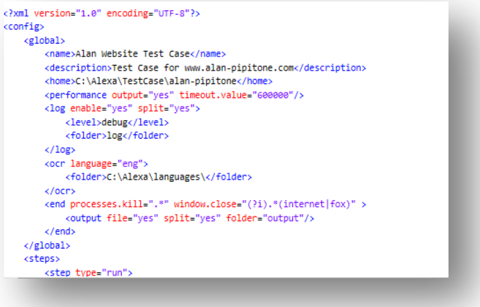


Look for technical solutions…

- Looking for a solution able to test the performance perceived by the users on applications published over Citrix, Teminal Server

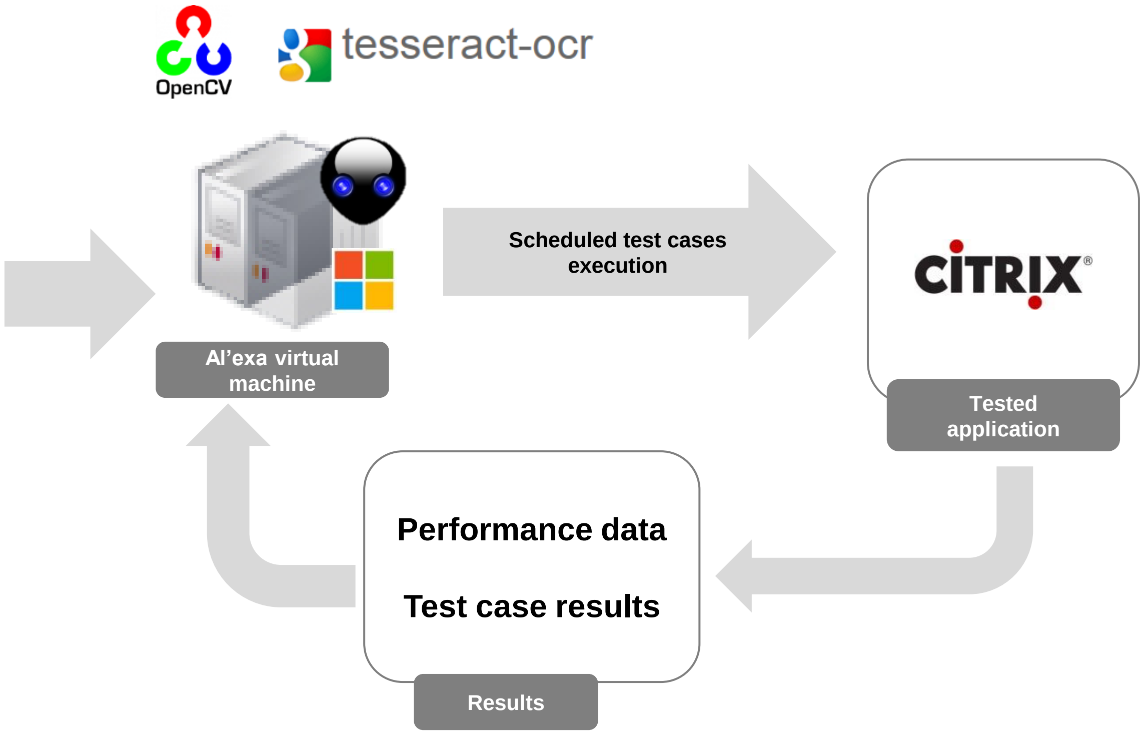
- Look for possible solutions
    - AUTOIT
    - OpenCV
    - Tesseract-ocr
    - Python
    - XML

The first Al’exa generation



```
<?xml version="1.0" encoding="UTF-8"?>
<config>
    <global>
        <name>Alan Website Test Case</name>
        <description>Test Case for www.alan-pipitone.com</description>
        <home>C:\Alexa\TestCase\alan-pipitone</home>
        <performance output="yes" timeout.value="600000"/>
        <log enable="yes" split="yes">
            <level>debug</level>
            <folder>log</folder>
        </log>
        <ocr language="eng">
            <folder>C:\Alexa\languages\</folder>
        </ocr>
        <end processes.kill=".*" window.close="(?i).*(internet|fox)" >
            <output file="yes" split="yes" folder="output"/>
        </end>
    </global>
    <steps>
        <step type="run">
```

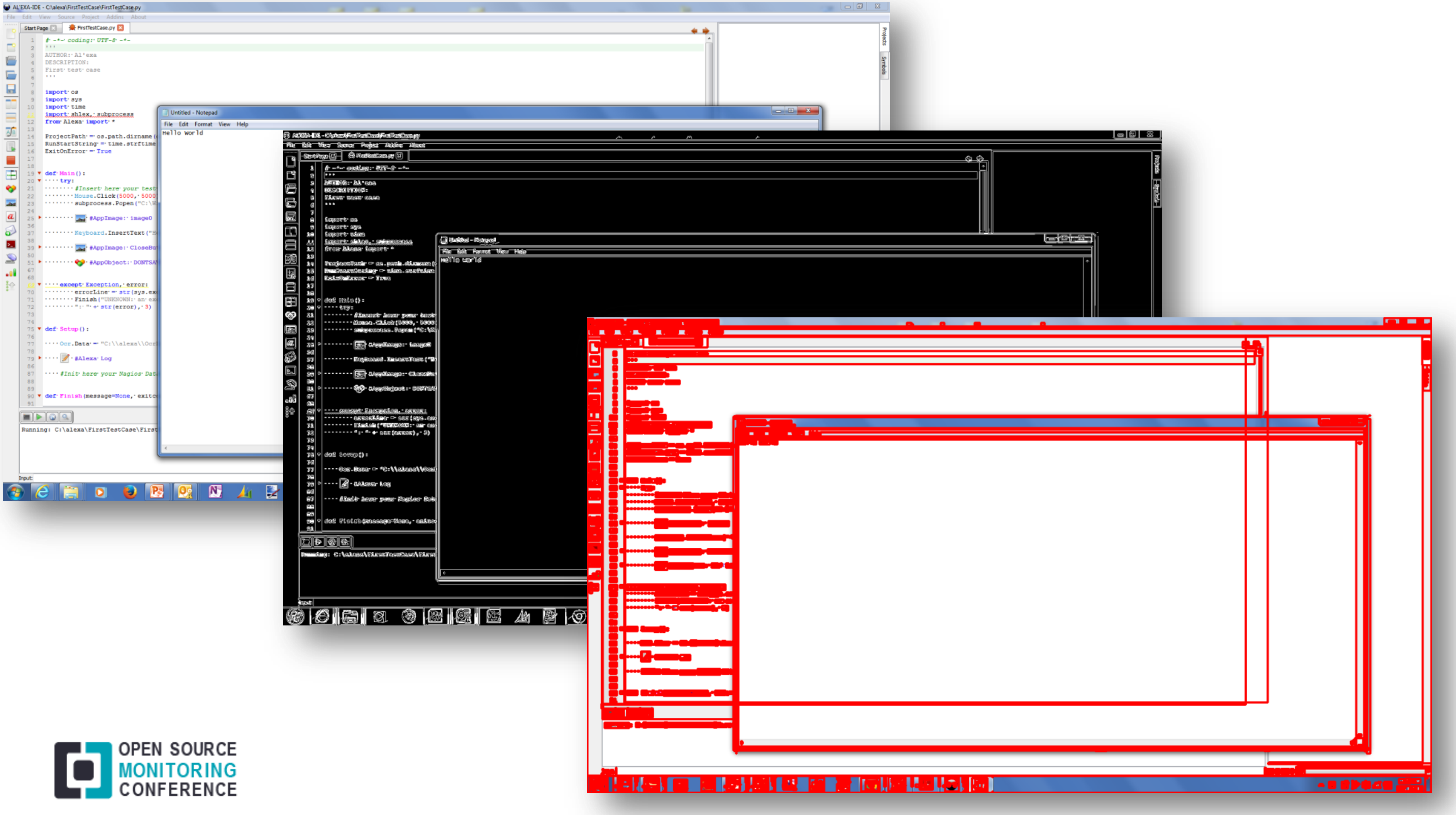
XML test cases



Performance data

Test case results

Al’exa vision with OpenCV



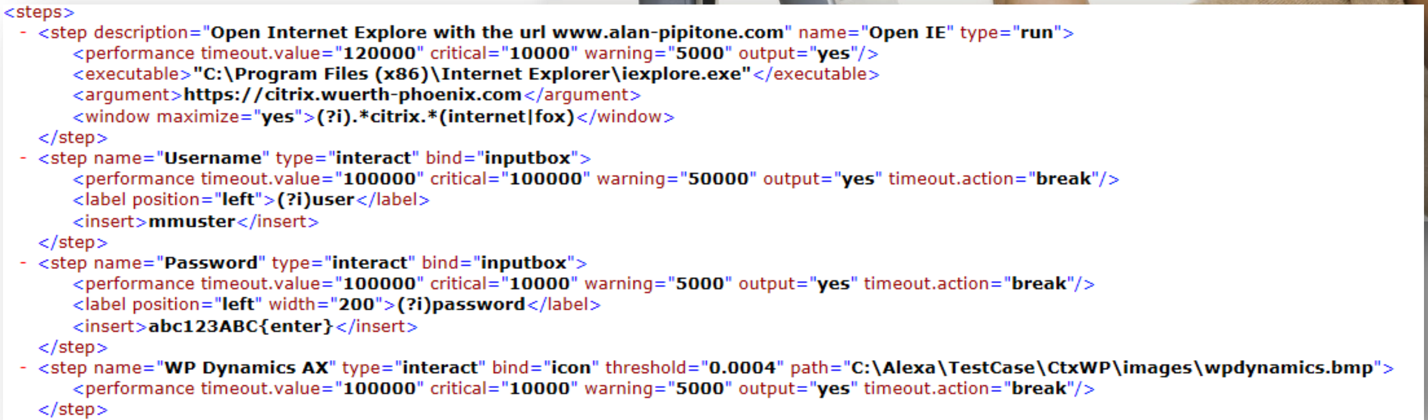
AL'EXA-IDE - C:\alexa\FirstTestCase\FirstTestCase.py
File Edit View Source Project Addins About
Start Page
FirstTestCase.py

```
# -*- coding: UTF-8 -*-
'''
AUTHOR: Al'exa
DESCRIPTION:
First test case
'''

import os
import sys
import time
import shlex, subprocess
from Alexa import *

ProjectPath = os.path.dirname(
RunStartString = time.strftime
ExitOnError = True


def Main():
    try:
        #Insert here your test
        Mouse.Click(5000, 5000
        subprocess.Popen("C:\W

        #AppImage: image0

        Keyboard.InsertText("H

        #AppImage: CloseBu

        #AppObject: DONTSA


    except Exception, error:
        errorLine = str(sys.ex
        Finish("UNKNOWN: an ex
        ": " + str(error), 3)


def Setup():

    Ocr.Data = "C:\\alexa\\Ocr

    #Alexa Log

    #Init here your Nagios Dat


def Finish(message=None, exitc
```

```
Running: C:\alexa\FirstTestCase\First
```

```
Hello World
```

The challenge of managing XML files



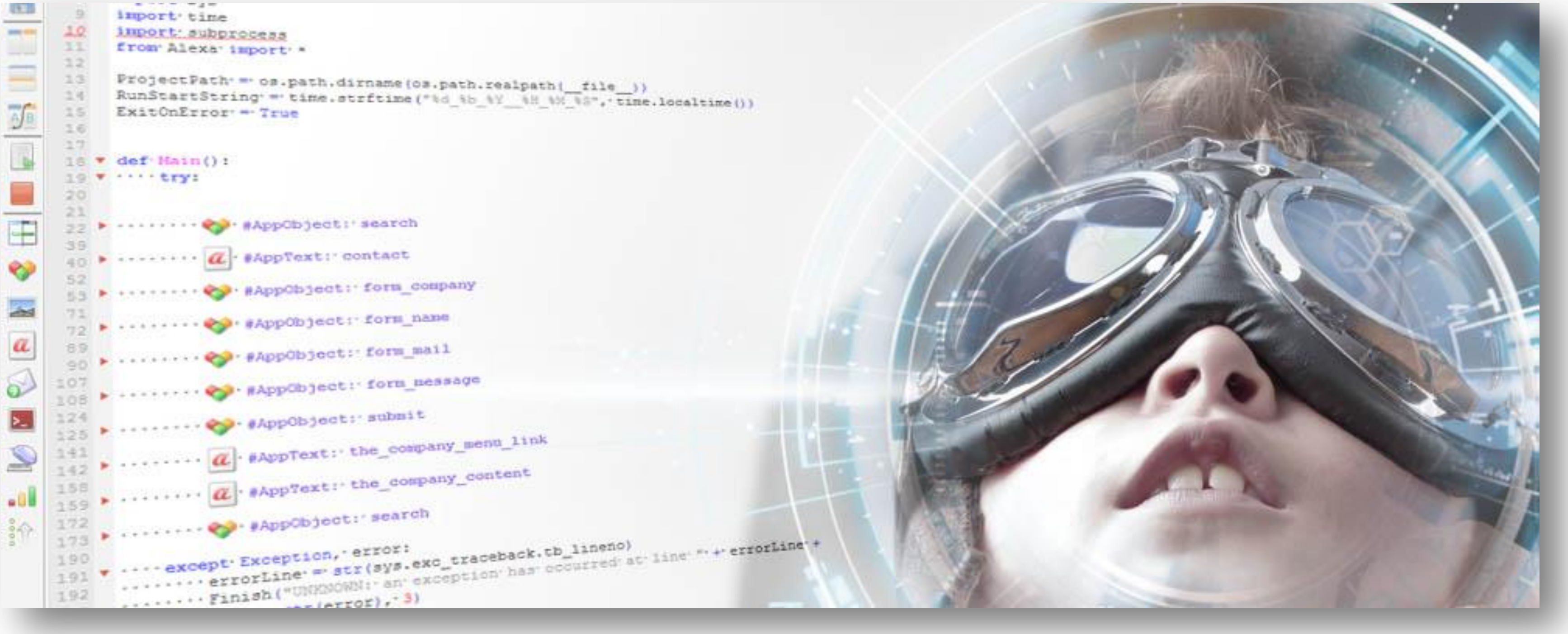
```
<steps>
- <step description="Open Internet Explore with the url www.alan-pipitone.com" name="Open IE" type="run">
      <performance timeout.value="120000" critical="10000" warning="5000" output="yes"/>
      <executable>"C:\Program Files (x86)\Internet Explorer\iexplore.exe"</executable>
      <argument>https://citrix.wuerth-phoenix.com</argument>
      <window maximize="yes">(?i).*citrix.*(internet|fox)</window>
  </step>
- <step name="Username" type="interact" bind="inputbox">
      <performance timeout.value="100000" critical="100000" warning="50000" output="yes" timeout.action="break"/>
      <label position="left">(?i)user</label>
      <insert>mmuster</insert>
  </step>
- <step name="Password" type="interact" bind="inputbox">
      <performance timeout.value="100000" critical="10000" warning="5000" output="yes" timeout.action="break"/>
      <label position="left" width="200">(?i)password</label>
      <insert>abc123ABC{enter}</insert>
  </step>
- <step name="WP Dynamics AX" type="interact" bind="icon" threshold="0.0004" path="C:\Alexa\TestCase\CtxWP\images\wpdynamics.bmp">
      <performance timeout.value="100000" critical="10000" warning="5000" output="yes" timeout.action="break"/>
  </step>
```

Drawbacks of XML files

- No IDE to create XML Files
- Difficulty to create and complexity to maintain large automation scenarios
- Poor possibility to handle applications exceptions
- Limited extensibility
  (i.e. Use data from an external database)

The second Al’exa generation

python + NINJA ide + Al’exa Plugin



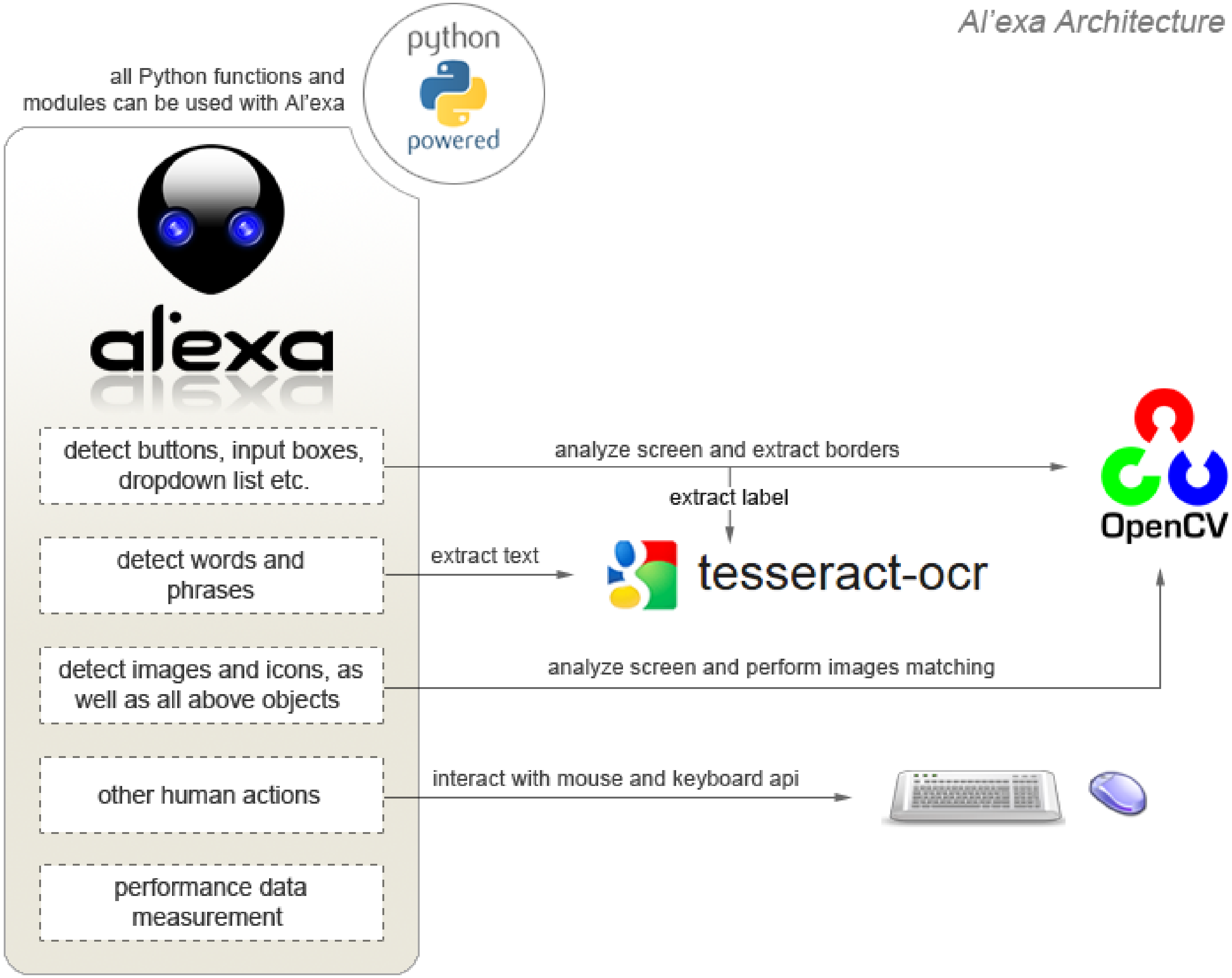
```
import time
import subprocess
from Alexa import *

ProjectPath = os.path.dirname(os.path.realpath(__file__))
RunStartString = time.strftime("%d_%b_%Y__%H_%M_%S", time.localtime())
ExitOnError = True


def Main():
    try:


        #AppObject: search
        #AppText: contact
        #AppObject: form_company
        #AppObject: form_name
        #AppObject: form_mail
        #AppObject: form_message
        #AppObject: submit
        #AppText: the_company_menu_link
        #AppText: the_company_content
        #AppObject: search
    except Exception, error:
        errorLine = str(sys.exc_traceback.tb_lineno)
        Finish("UNKNOWN: an exception has occurred at line " + errorLine +
               str(error), 3)
```

alexa

WÜRTHPHOENIX

The new architecture

Al'exa Architecture



all Python functions and modules can be used with Al'exa

python powered

alexa

detect buttons, input boxes, dropdown list etc.

detect words and phrases

detect images and icons, as well as all above objects

other human actions

performance data measurement

analyze screen and extract borders

extract label

extract text

tesseract-ocr

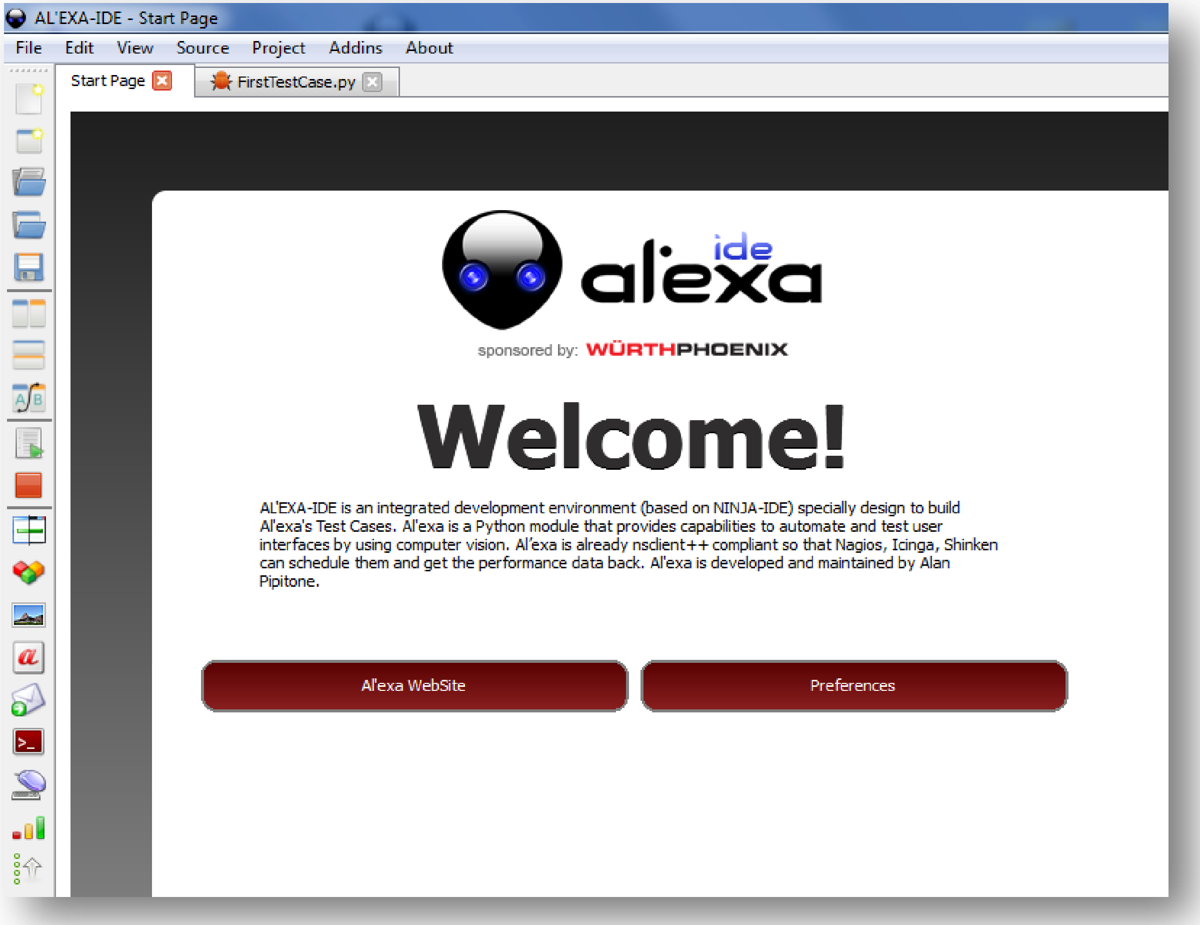
OpenCV

analyze screen and perform images matching

interact with mouse and keyboard api

OPEN SOURCE MONITORING CONFERENCE

Creation of test cases with Al’exa IDE

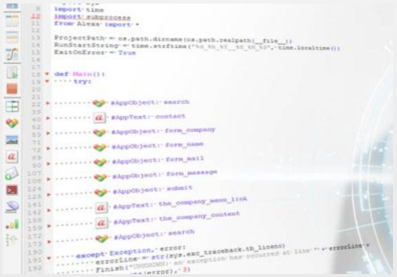


Welcome!

AL'EXA-IDE is an integrated development environment (based on NINJA-IDE) specially design to build Al'exa's Test Cases. Al'exa is a Python module that provides capabilities to automate and test user interfaces by using computer vision. Al'exa is already nsclient++ compliant so that Nagios, Icinga, Shinken can schedule them and get the performance data back. Al'exa is developed and maintained by Alan Pipitone.

Al‘exa IDE

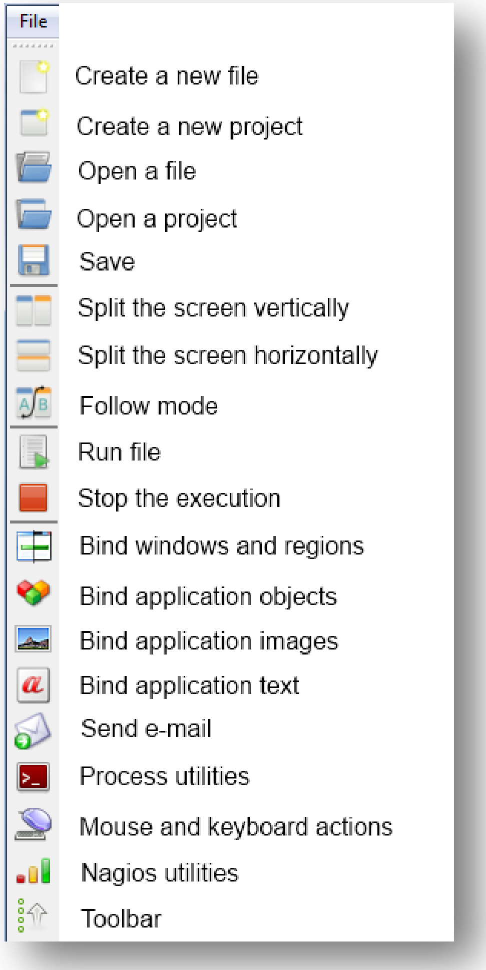
- Wizard for the creation of test cases
- Possibility to create complex check logic thanks to the integration with Phyton



```
import time
import subprocess
from Alexa import *

ProjectPath = os.path.dirname(os.path.realpath(__file__))
RunStartString = time.strftime("%d_%m_%Y__%H_%M_%S", time.localtime())
ExitOnError = True


def Main():
    try:

        #AppObject: search
        #AppText: contact
        #AppObject: form_company
        #AppObject: form_name
        #AppObject: form_mail
        #AppObject: form_message
        #AppObject: submit
        #AppText: the_company_menu_link
        #AppText: the_company_content
        #AppObject: search
    except Exception, error:
        errorLine = str(sys.exc_traceback.tb_lineno)
        Finish("UNKNOWN: an exception has occurred at line " + errorLine + str(error), 3)
```

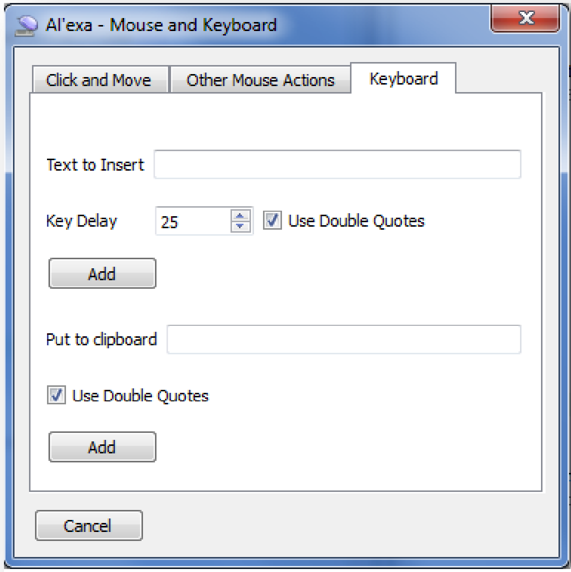
OPEN SOURCE MONITORING CONFERENCE

Toolbar and interaction with mouse and keyboard



File

- Create a new file
- Create a new project
- Open a file
- Open a project
- Save
- Split the screen vertically
- Split the screen horizontally
- Follow mode
- Run file
- Stop the execution
- Bind windows and regions
- Bind application objects
- Bind application images
- Bind application text
- Send e-mail
- Process utilities
- Mouse and keyboard actions
- Nagios utilities
- Toolbar



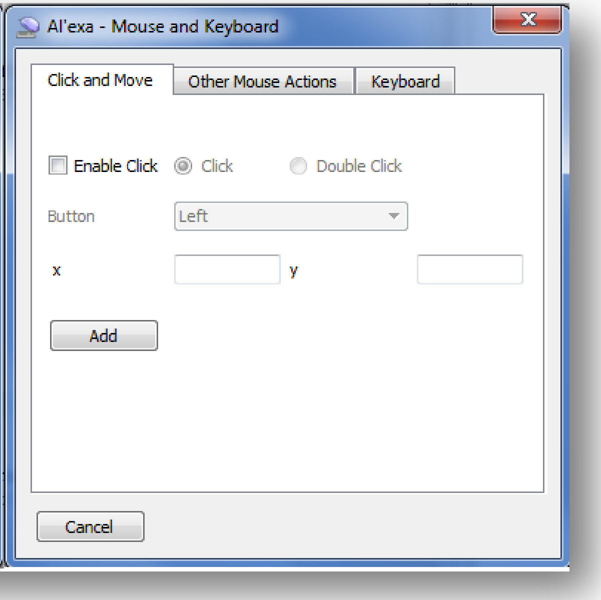
Al'exa - Mouse and Keyboard

Text to Insert:

Key Delay: 25 ☑ Use Double Quotes

Put to clipboard:

☑ Use Double Quotes



Al'exa - Mouse and Keyboard

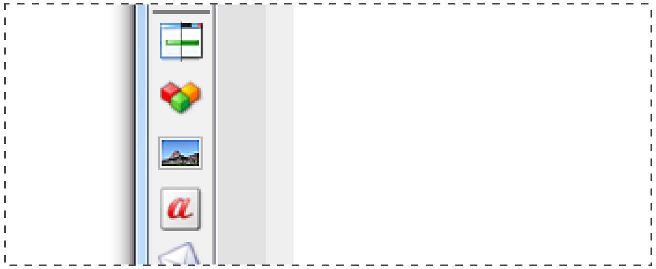
□ Enable Click ● Click ○ Double Click

Button: Left

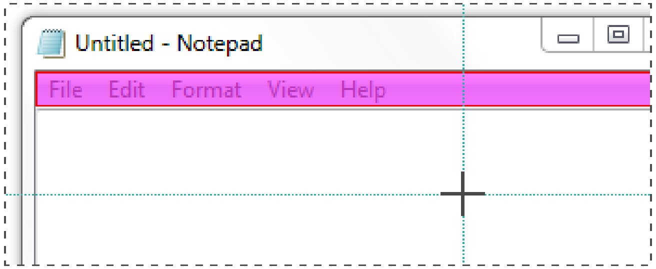
x: y:

Easy creation of test cases through the IDE

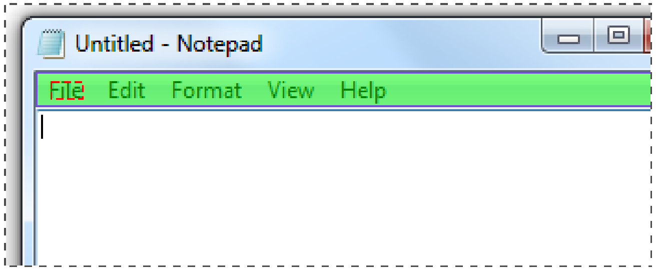
1. *Toolbar to easily add Al'exa Classes and Methods*



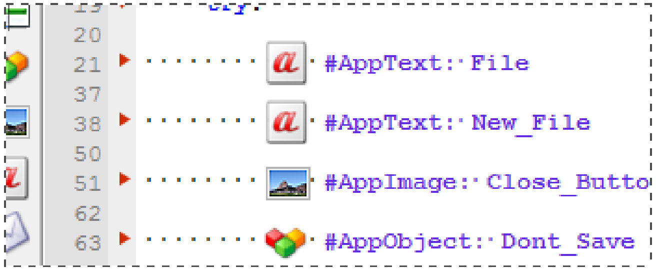
2. *Advanced GUI to easily identify Application Objects*



3. *Powerful Tools to test Al'exa behavior*



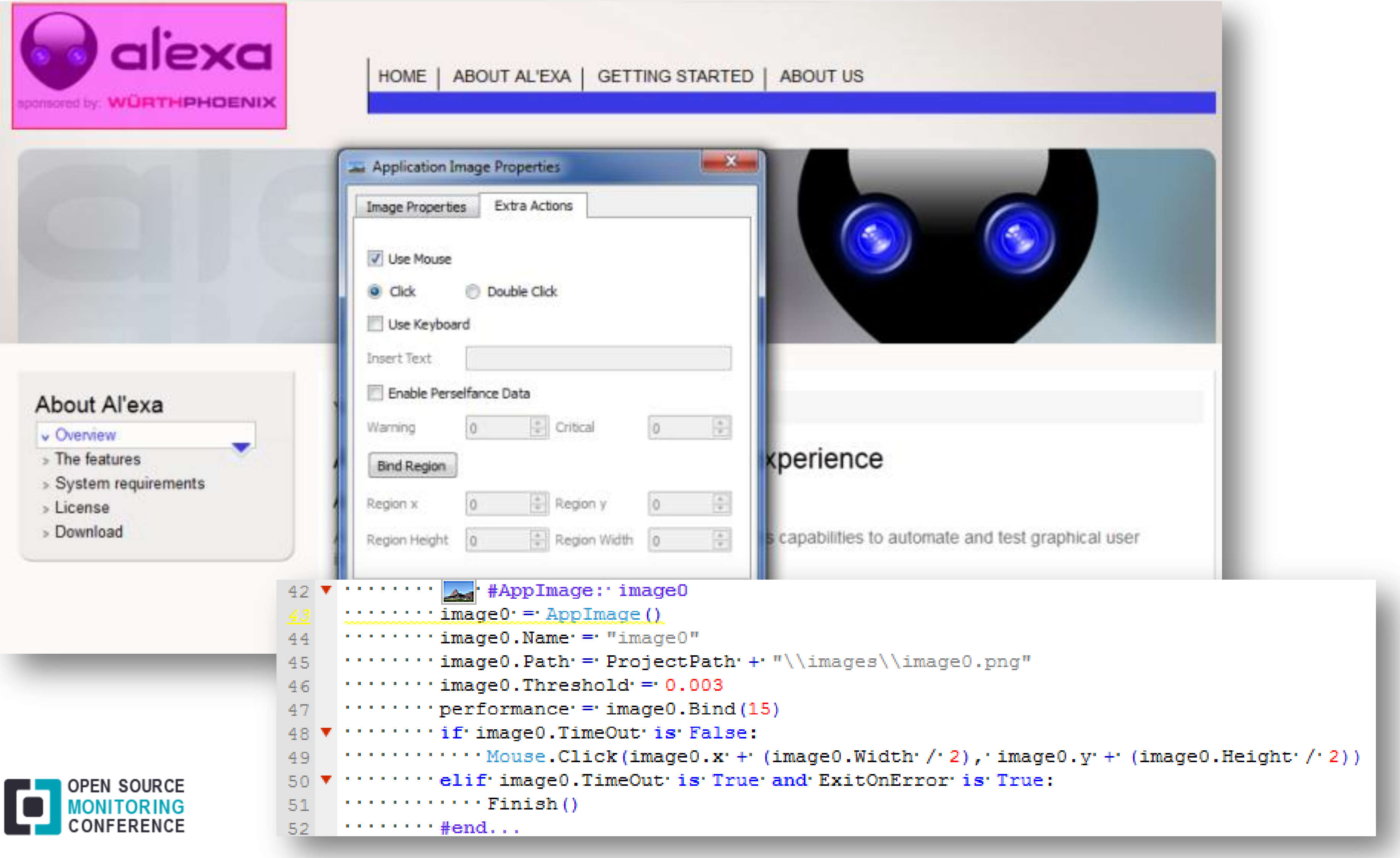
4. *Automatic Code Generator*



```
        #AppText::File
        #AppText::New_File
        #AppImage::Close_Butto
        #AppObject::Dont_Save
```

5. And much more…

Bind an image


HOME | ABOUT AL'EXA | GETTING STARTED | ABOUT US
About Al'exa
Overview
The features
System requirements
License
Download
xperience
s capabilities to automate and test graphical user

Application Image Properties

Image Properties | Extra Actions

☑ Use Mouse

● Click ○ Double Click

□ Use Keyboard

Insert Text

□ Enable Perselfance Data

Warning 0 Critical 0

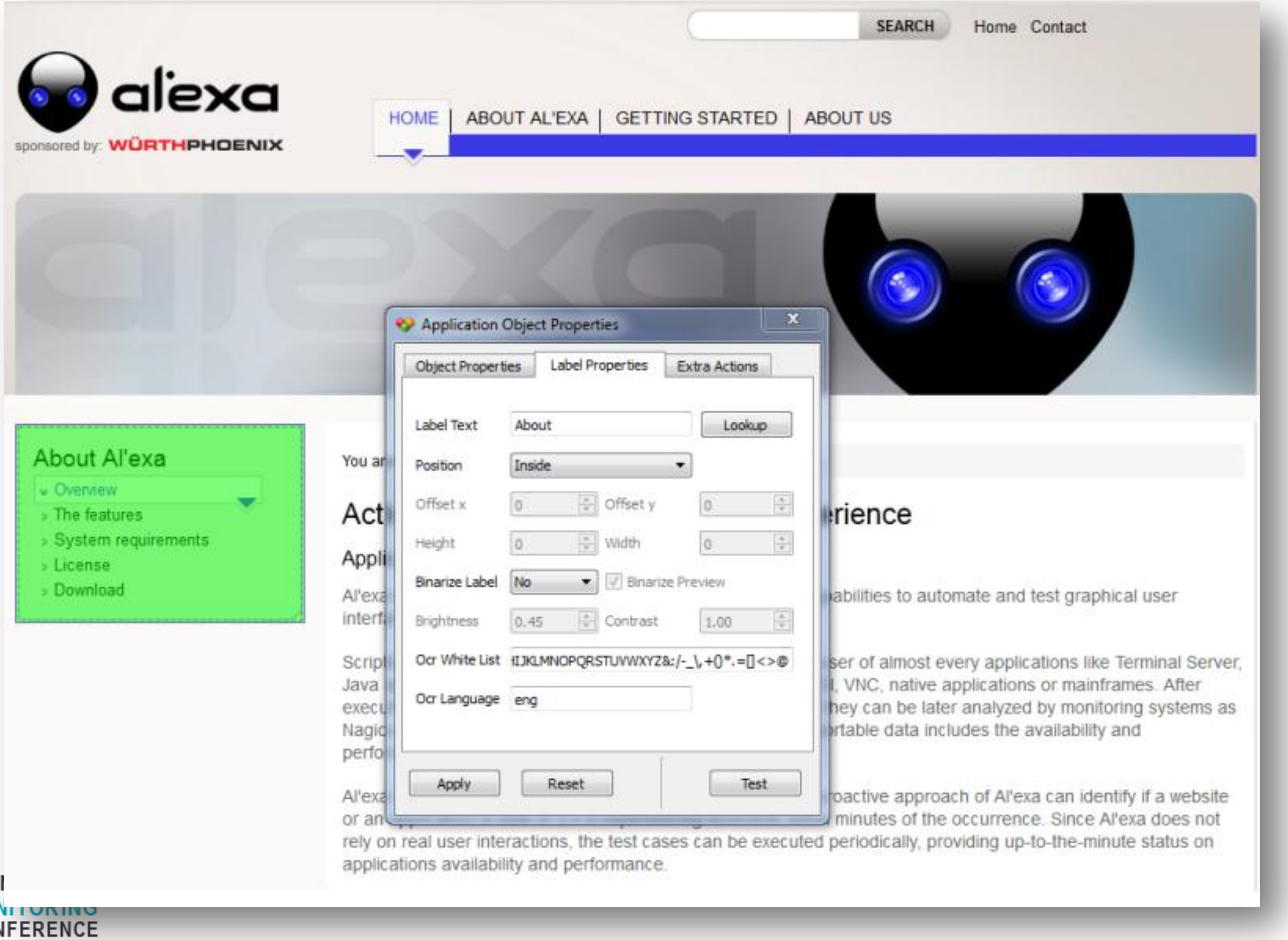
Bind Region

Region x 0 Region y 0

Region Height 0 Region Width 0

```
        #AppImage: image0
        image0 = AppImage()
        image0.Name = "image0"
        image0.Path = ProjectPath + "\\images\\image0.png"
        image0.Threshold = 0.003
        performance = image0.Bind(15)
        if image0.TimeOut is False:
            Mouse.Click(image0.x + (image0.Width / 2), image0.y + (image0.Height / 2))
        elif image0.TimeOut is True and ExitOnError is True:
            Finish()
        #end...
```

Bind an object



Label Text: About

Position: Inside

Offset x: 0 Offset y: 0

Height: 0 Width: 0

Binarize Label: No ☑ Binarize Preview

Brightness: 0.45 Contrast: 1.00

Ocr White List: IJKLMNOPQRSTUVWXYZ&:/-_\,+()*.=[]<>@

Ocr Language: eng

You ar

Act erience

Appli

Al'exa abilities to automate and test graphical user interf

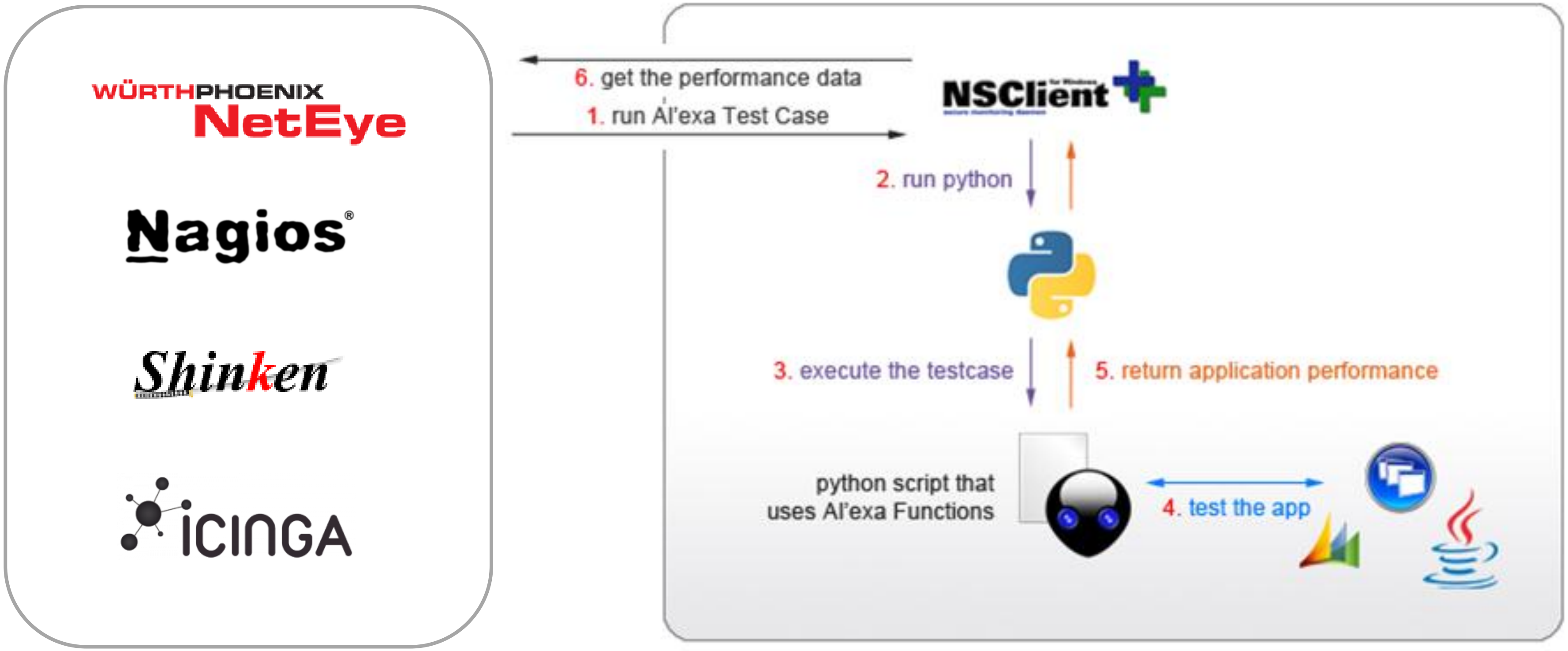
Scrip ser of almost every applications like Terminal Server, Java , VNC, native applications or mainframes. After execu hey can be later analyzed by monitoring systems as Nagio rtable data includes the availability and perfo

Al'exa roactive approach of Al'exa can identify if a website or an minutes of the occurrence. Since Al'exa does not rely on real user interactions, the test cases can be executed periodically, providing up-to-the-minute status on applications availability and performance.

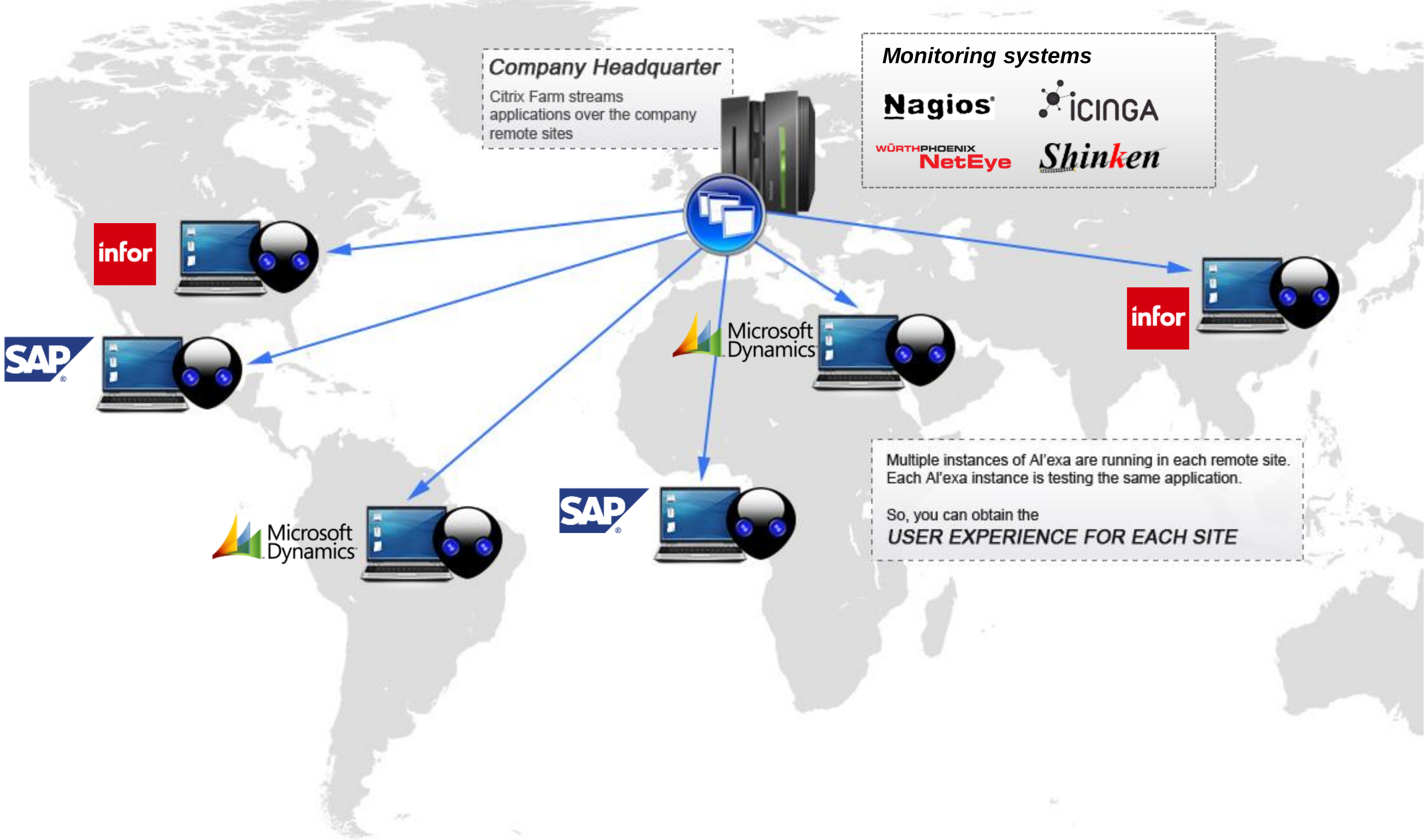
The monitoring system integration

Monitoring systems

Al'exa virtual machine



Distributed testing of a Citrix farm



Company Headquarter

Citrix Farm streams applications over the company remote sites

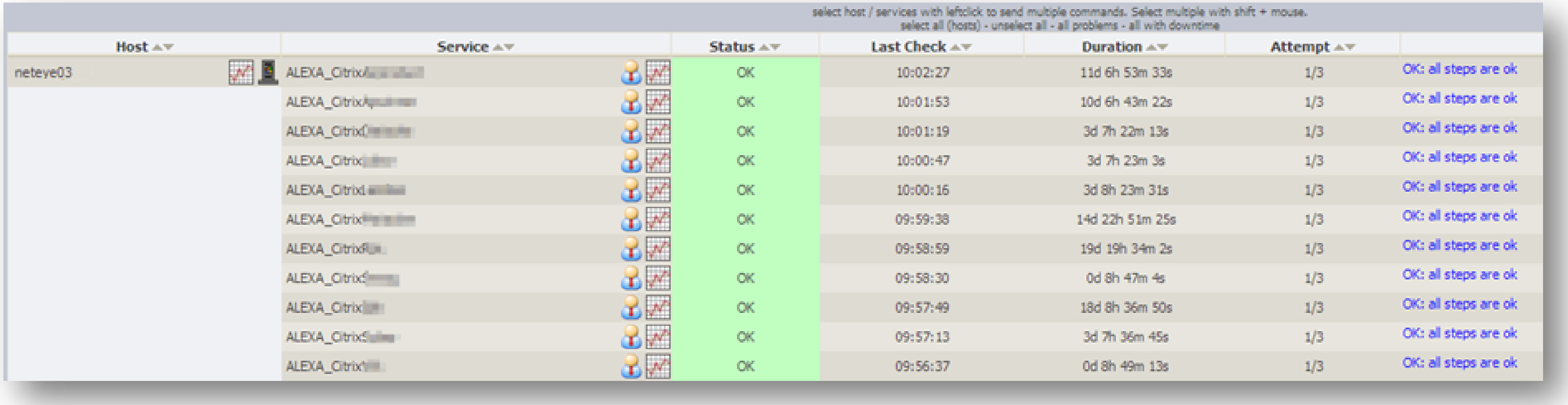
Monitoring systems

Multiple instances of Al'exa are running in each remote site. Each Al'exa instance is testing the same application.

So, you can obtain the

USER EXPERIENCE FOR EACH SITE

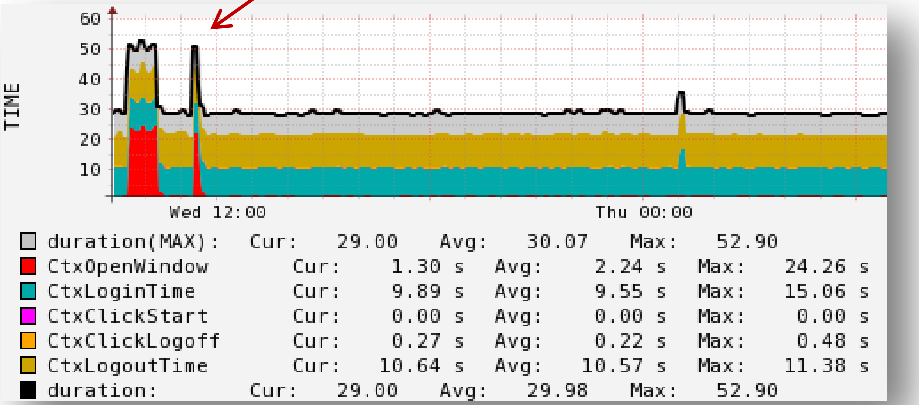
Performance statistics through monitoring systems



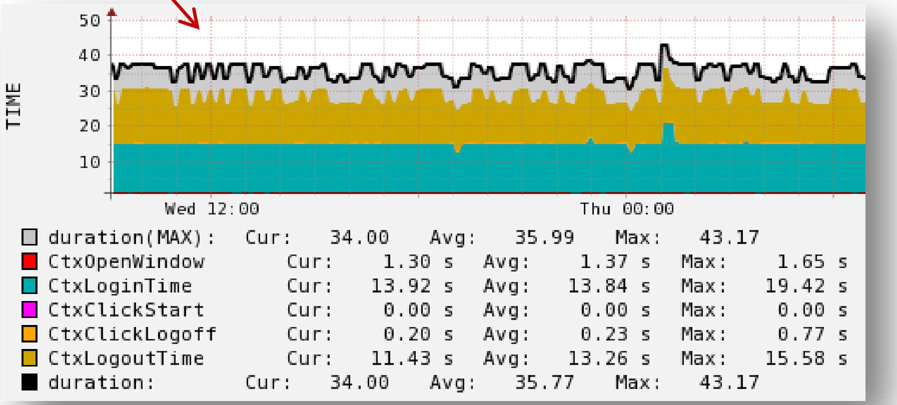
select host / services with leftclick to send multiple commands. Select multiple with shift + mouse.
select all (hosts) - unselect all - all problems - all with downtime

Host	Service	Status	Last Check	Duration	Attempt	
neteye03	ALEXA_Citrix[redacted]	OK	10:02:27	11d 6h 53m 33s	1/3	OK: all steps are ok
	ALEXA_Citrix[redacted]	OK	10:01:53	10d 6h 43m 22s	1/3	OK: all steps are ok
	ALEXA_Citrix[redacted]	OK	10:01:19	3d 7h 22m 13s	1/3	OK: all steps are ok
	ALEXA_Citrix[redacted]	OK	10:00:47	3d 7h 23m 3s	1/3	OK: all steps are ok
	ALEXA_Citrix[redacted]	OK	10:00:16	3d 8h 23m 31s	1/3	OK: all steps are ok
	ALEXA_Citrix[redacted]	OK	09:59:38	14d 22h 51m 25s	1/3	OK: all steps are ok
	ALEXA_Citrix[redacted]	OK	09:58:59	19d 19h 34m 2s	1/3	OK: all steps are ok
	ALEXA_Citrix[redacted]	OK	09:58:30	0d 8h 47m 4s	1/3	OK: all steps are ok
	ALEXA_Citrix[redacted]	OK	09:57:49	18d 8h 36m 50s	1/3	OK: all steps are ok
	ALEXA_Citrix[redacted]	OK	09:57:13	3d 7h 36m 45s	1/3	OK: all steps are ok
	ALEXA_Citrix[redacted]	OK	09:56:37	0d 8h 49m 13s	1/3	OK: all steps are ok

Same check performed
from different access point



```
□ duration(MAX):   Cur:   29.00    Avg:   30.07    Max:   52.90
■ CtxOpenWindow        Cur:    1.30 s   Avg:    2.24 s   Max:   24.26 s
■ CtxLoginTime         Cur:    9.89 s   Avg:    9.55 s   Max:   15.06 s
■ CtxClickStart        Cur:    0.00 s   Avg:    0.00 s   Max:    0.00 s
■ CtxClickLogoff       Cur:    0.27 s   Avg:    0.22 s   Max:    0.48 s
■ CtxLogoutTime        Cur:   10.64 s   Avg:   10.57 s   Max:   11.38 s
■ duration:        Cur:   29.00    Avg:   29.98    Max:   52.90
```



```
□ duration(MAX):   Cur:   34.00    Avg:   35.99    Max:   43.17
■ CtxOpenWindow        Cur:    1.30 s   Avg:    1.37 s   Max:    1.65 s
■ CtxLoginTime         Cur:   13.92 s   Avg:   13.84 s   Max:   19.42 s
■ CtxClickStart        Cur:    0.00 s   Avg:    0.00 s   Max:    0.00 s
■ CtxClickLogoff       Cur:    0.20 s   Avg:    0.23 s   Max:    0.77 s
■ CtxLogoutTime        Cur:   11.43 s   Avg:   13.26 s   Max:   15.58 s
■ duration:        Cur:   34.00    Avg:   35.77    Max:   43.17
```

The major benefits of the solution

- Testing the reliability and availability of the business applications
- Collecting performance data of almost every applications on single user action
- Flexible testing: applications objects are recognized also in case of changes on the position, size or color.

Supported applications

CiTRIX

Java

VNC

vmware

SAP

Microsoft Dynamics AX

Adobe Flash Player

… or every application with a user interface

The features

- Creation of test cases for every kind of application that provides a User Interface, through the simulation of the user interactions
- Automatic recognition of all the application objects (input box, dropdownlist, icon, button, scrollbar)
- Automatic recognition of text in different languages (English, German, Italian, Korean, Japanese, check the entire list)
- Recording of the interactions with mouse and keyboards
- Easy and quick creation of test cases through the intuitive user interface of the Al'exa IDE
- Measurement of the performance data of every applications and check the availability of the IT services
- Management of the timeouts
- Debug modality and possibility to save the screenshots of the errors for further analysis
- Notifications via email with the error screenshots
- Possibility to set thresholds to generate critical and warning alerts
- Possibility to modify the test cases generate by Al'exa IDE by using Phyton as programming language
- Page snapshots when problems occur, allowing to identify the root cause of the error
- Possibility to send the performance data provided by Al'exa to monitoring systems as Nagios, Icinga, Shinken or WÜRTHPHOENIX NetEye, to generate reports and praphs for uptime and performance trends

Future evolving steps

- Retrieve all application published via Citrix and automatically execute the applications to check their availability

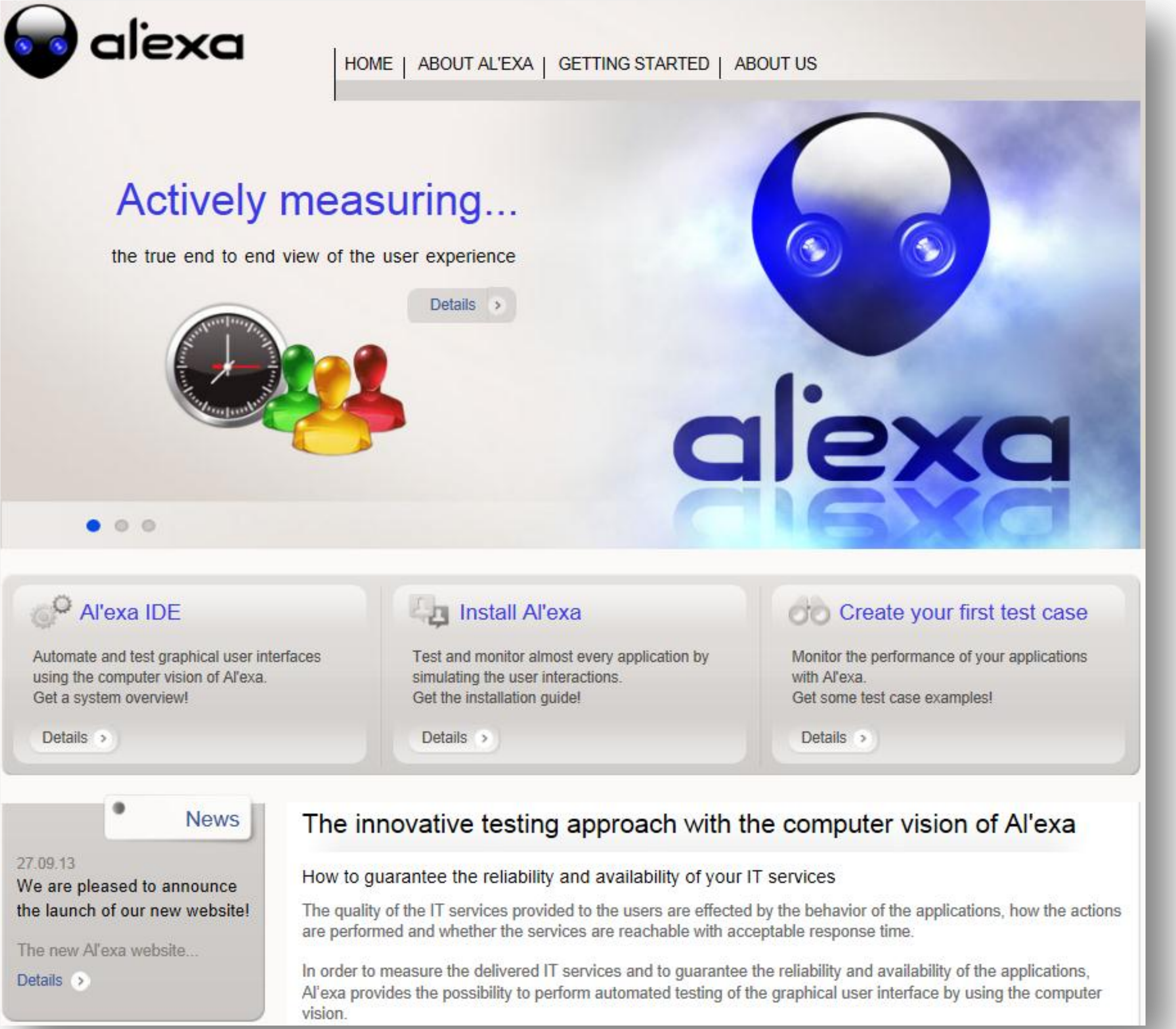
- Automate more actions and operations through the IDE, (i.e. method to record the user activity and create the code)

- Enhance some methods to further increase the flexibility of Al'exa

- Create an advanced module able to manage and troubleshoot application errors. (i.e. Allow Al'exa to automatically handle various exceptions)

- Create a complete artificial intelligence that can be easily trained.

…for more information www.alexa-monitoring.com



HOME | ABOUT AL'EXA | GETTING STARTED | ABOUT US

Actively measuring...

the true end to end view of the user experience

Details

Al'exa IDE

Automate and test graphical user interfaces using the computer vision of Al'exa.
Get a system overview!

Details

Install Al'exa

Test and monitor almost every application by simulating the user interactions.
Get the installation guide!

Details

Create your first test case

Monitor the performance of your applications with Al'exa.
Get some test case examples!

Details

News

27.09.13
We are pleased to announce the launch of our new website!

The new Al'exa website...

Details

The innovative testing approach with the computer vision of Al'exa

How to guarantee the reliability and availability of your IT services

The quality of the IT services provided to the users are effected by the behavior of the applications, how the actions are performed and whether the services are reachable with acceptable response time.

In order to measure the delivered IT services and to guarantee the reliability and availability of the applications, Al'exa provides the possibility to perform automated testing of the graphical user interface by using the computer vision.

OPEN SOURCE MONITORING CONFERENCE