

# TO $\infty$ AND BEYOND

... MAKING JUBULA SURPASS ITS LIMITS

by @MarkusTiede - BREDEX GmbH

# AGENDA

What's Jubula?

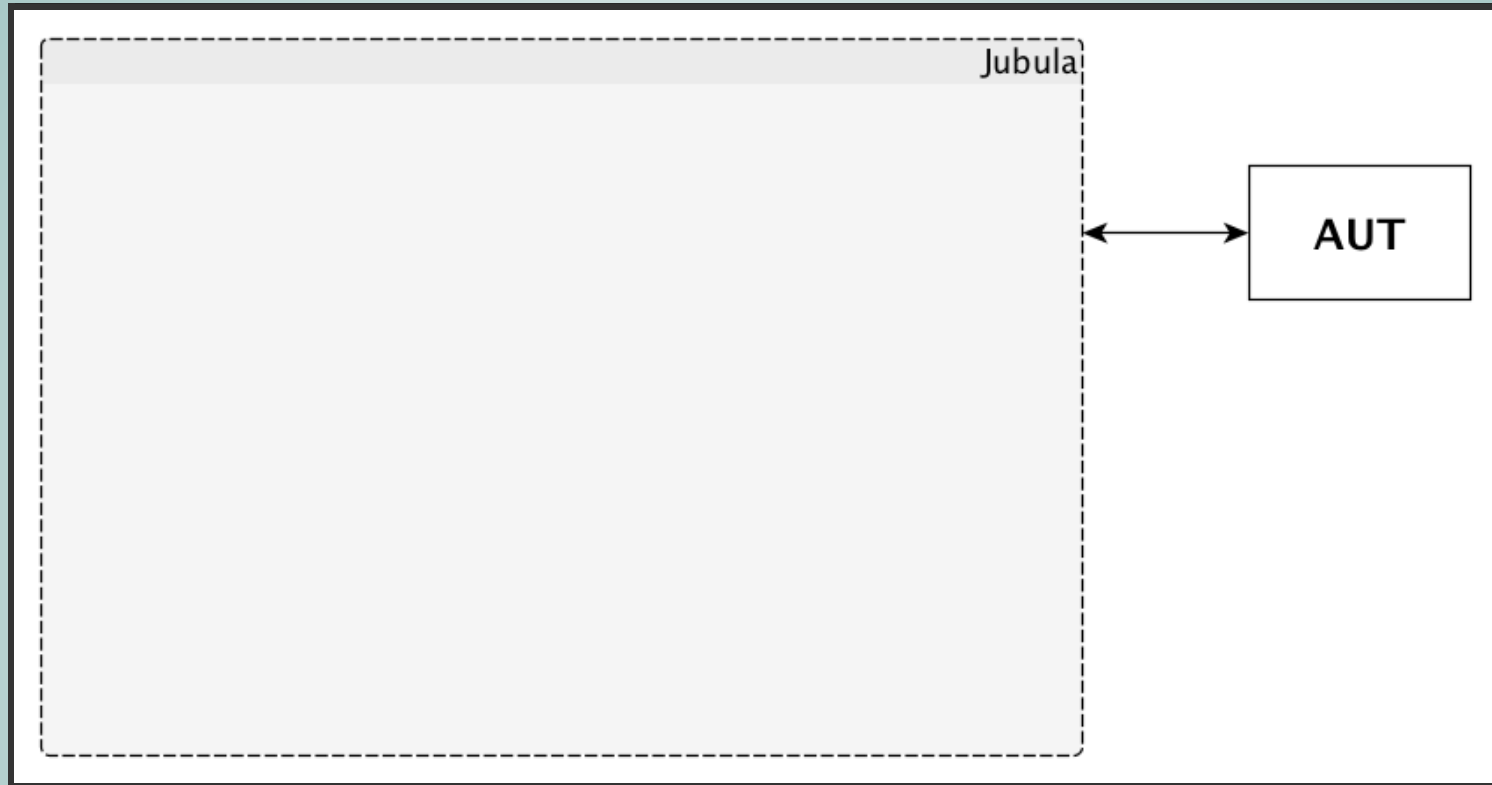
How & where to extend it?

# THAT'S JUBULA!

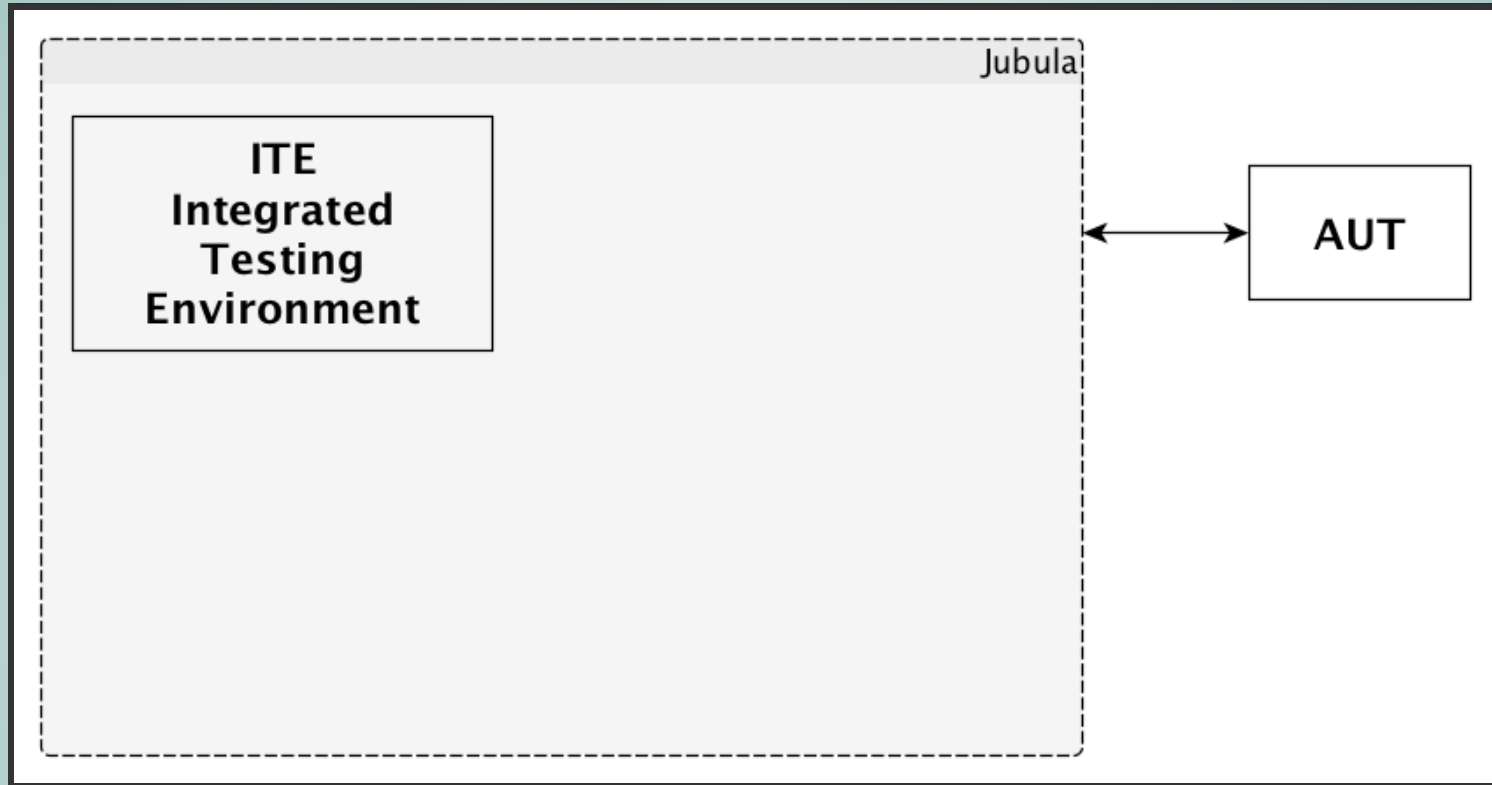


UI automation @ [eclipse.org/jubula](https://eclipse.org/jubula) since 2011

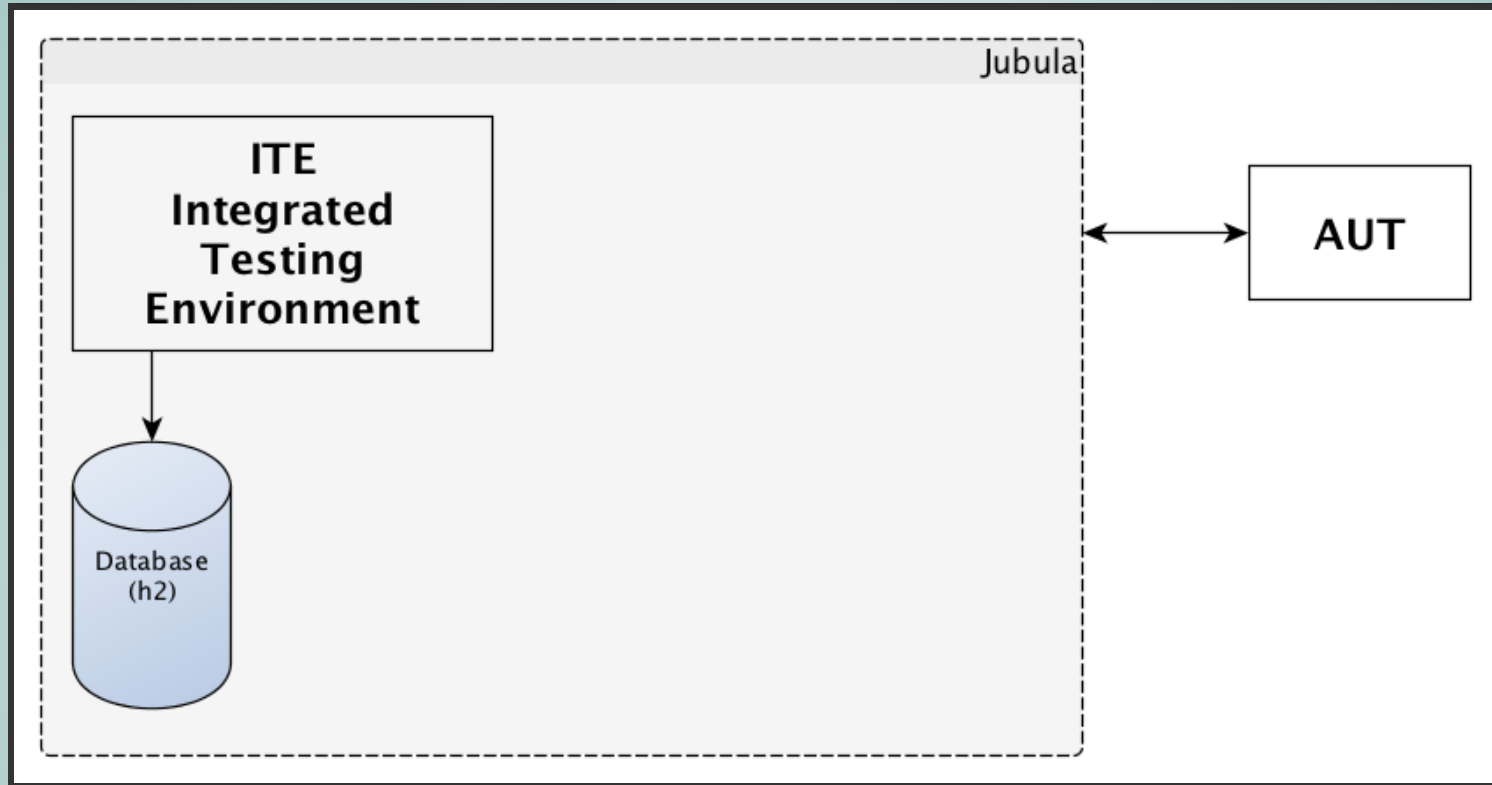
# JUBULA - THE BIG PICTURE



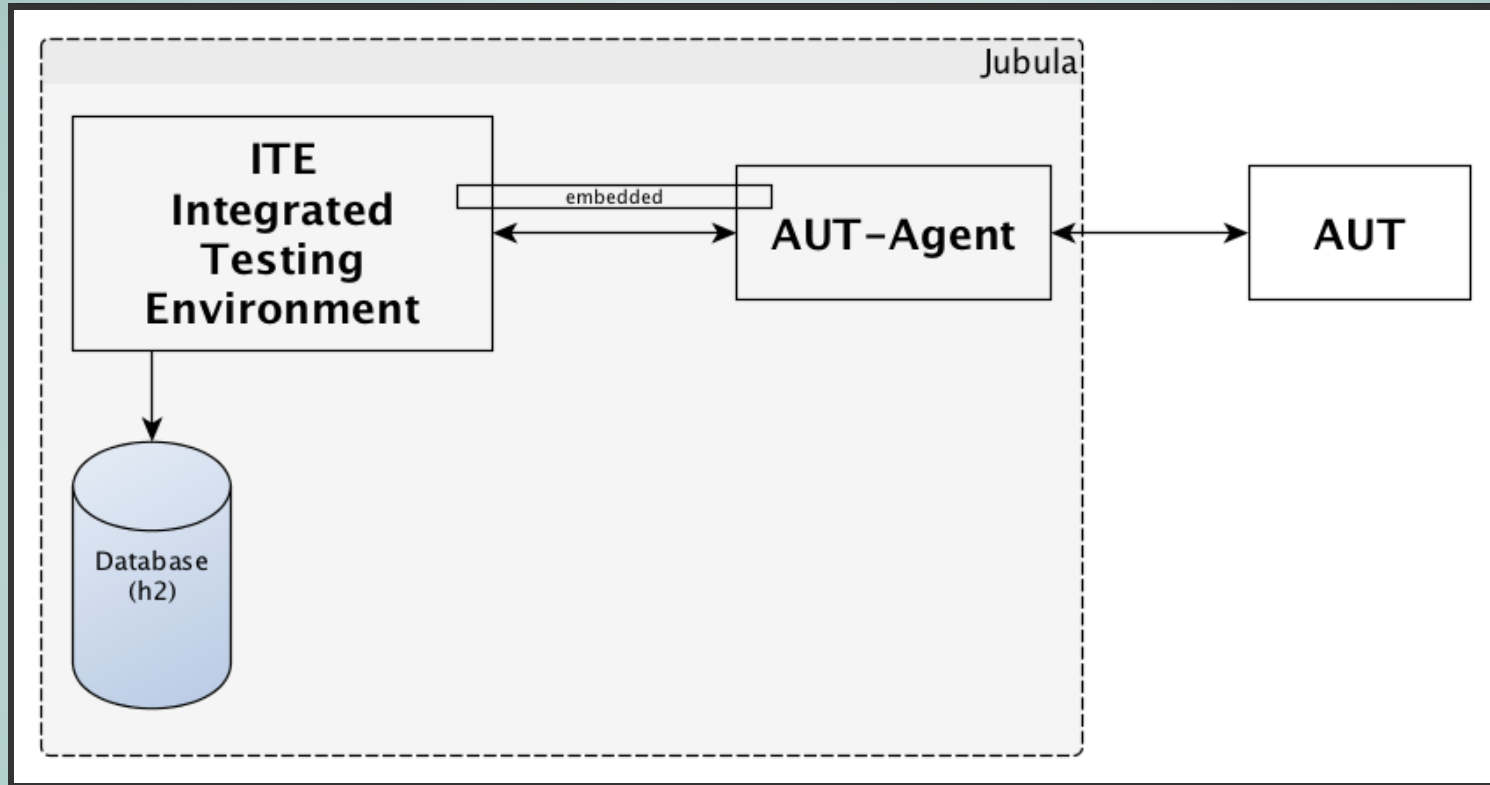
# JUBULA - THE BIG PICTURE



# JUBULA - THE BIG PICTURE

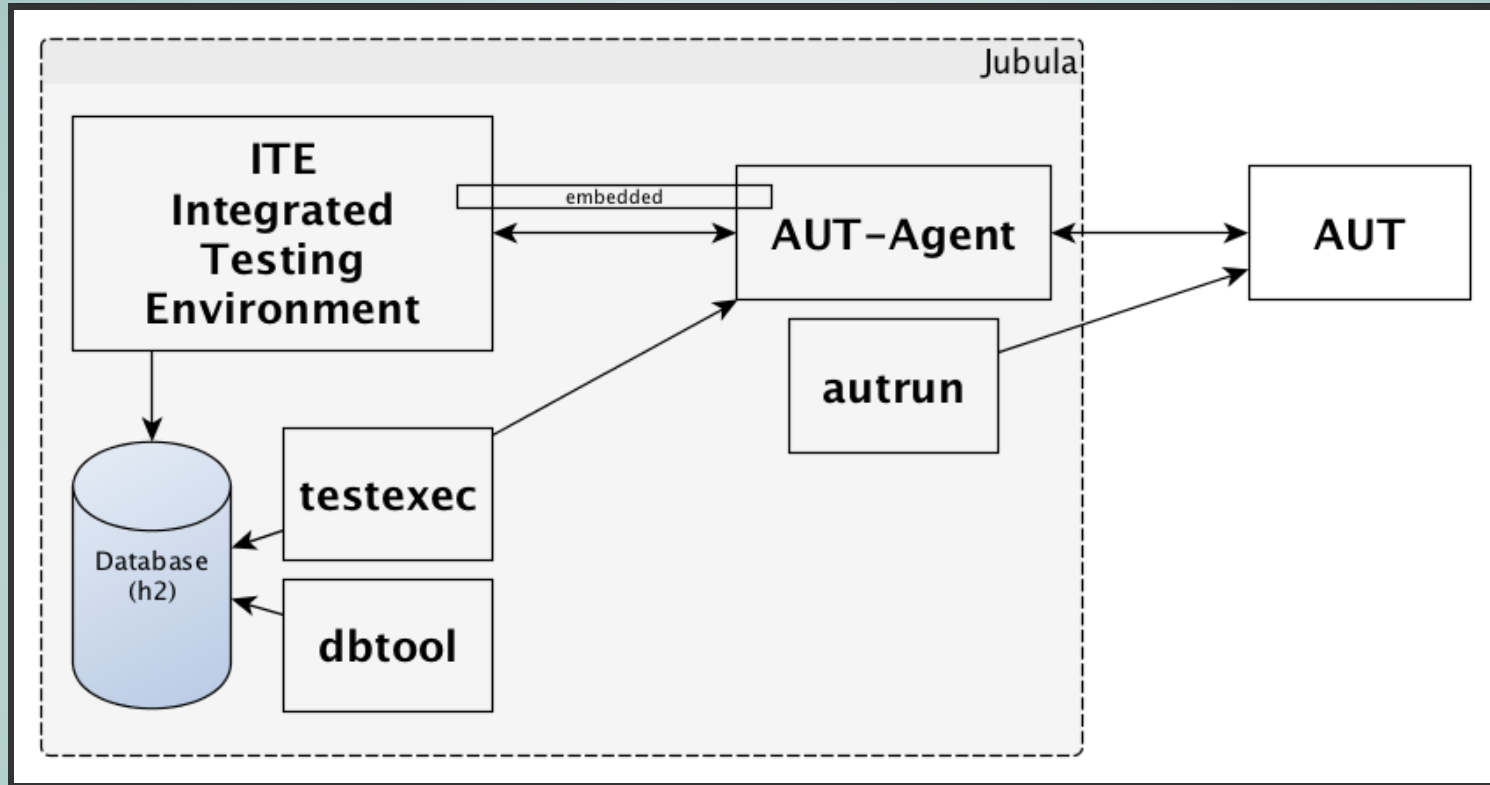


# JUBULA - THE BIG PICTURE





# JUBULA - THE BIG PICTURE

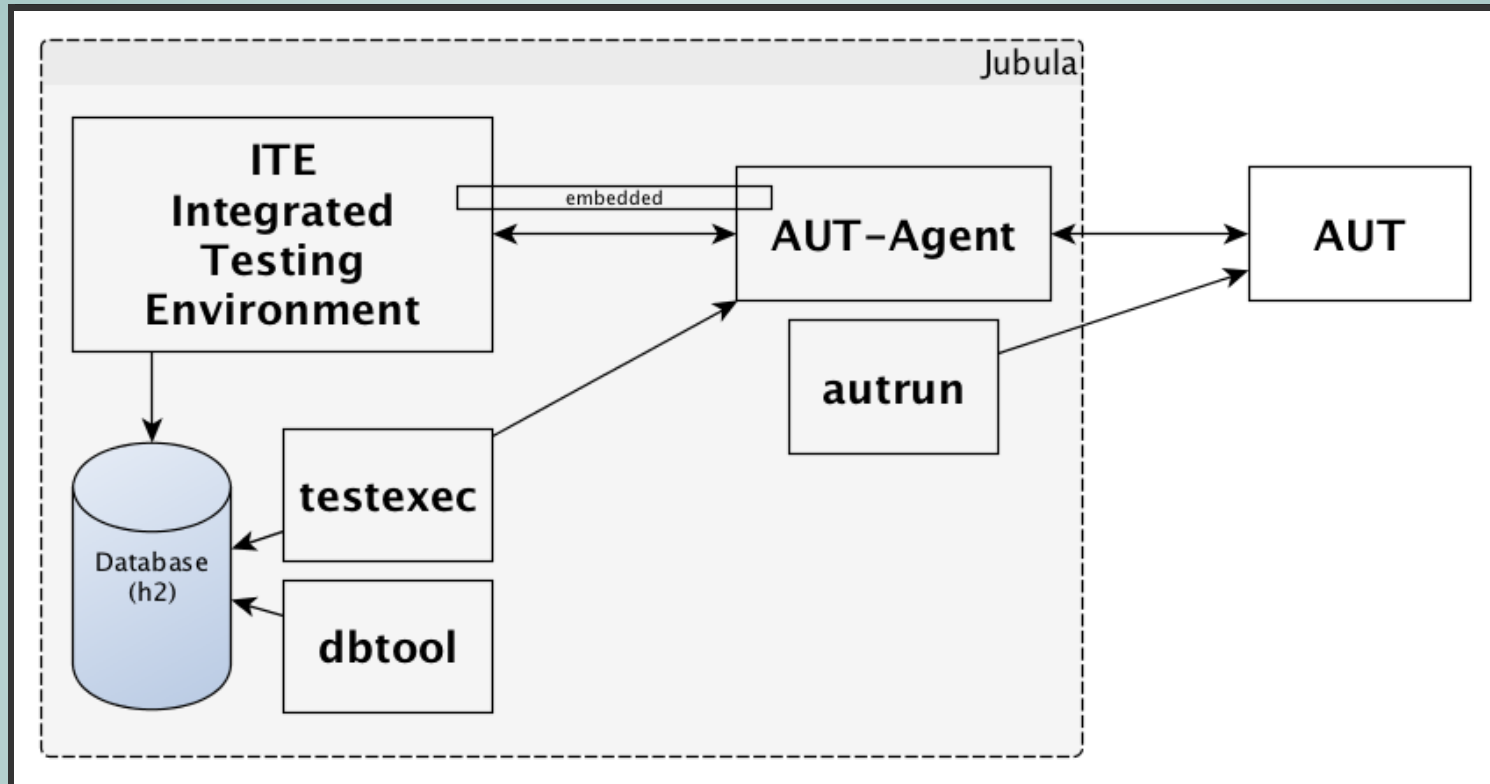




# DISCLAIMER

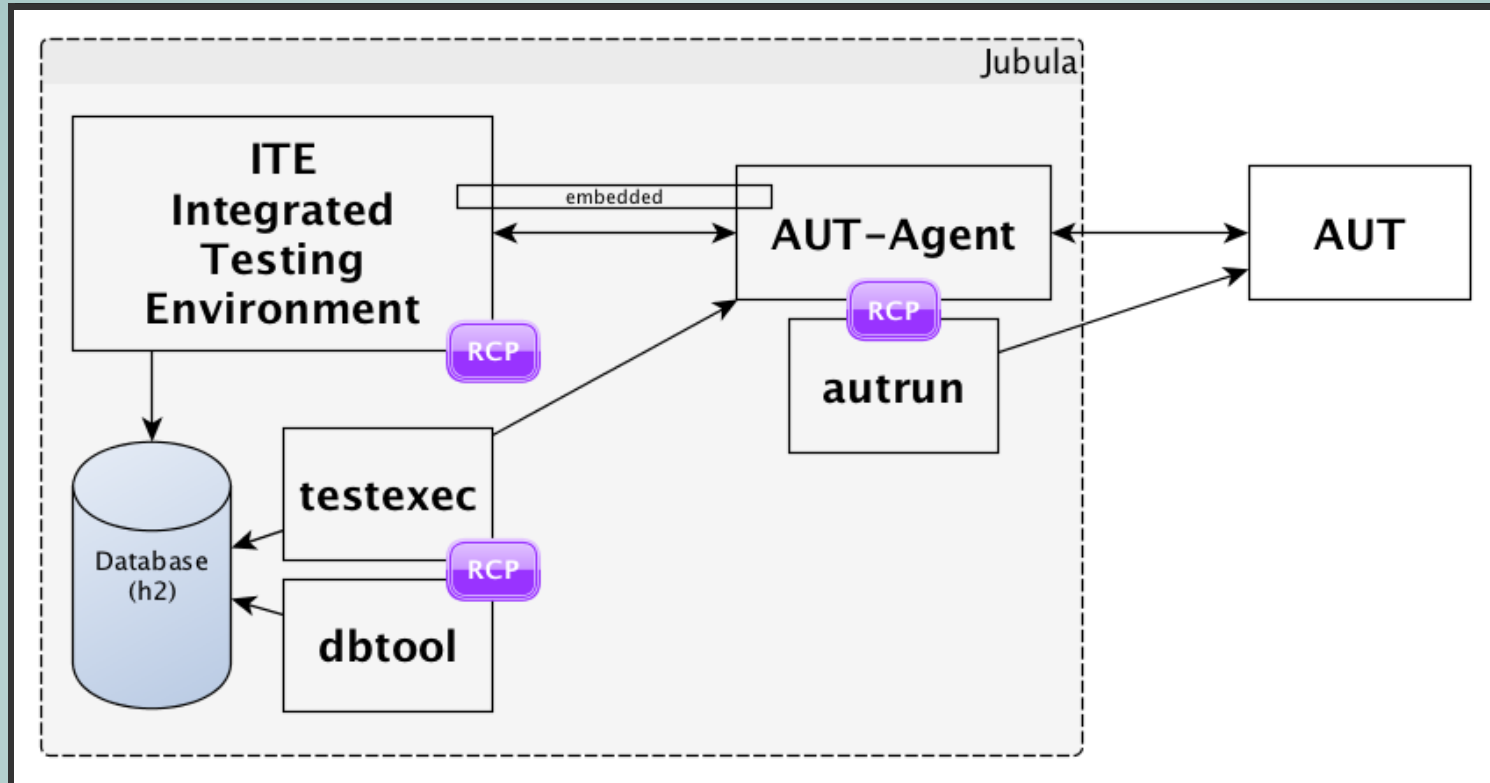
Testing with Jubula **does not** require programming skills  
... but extending Jubula **does** require programming skills!

# EXTENDING JUBULA



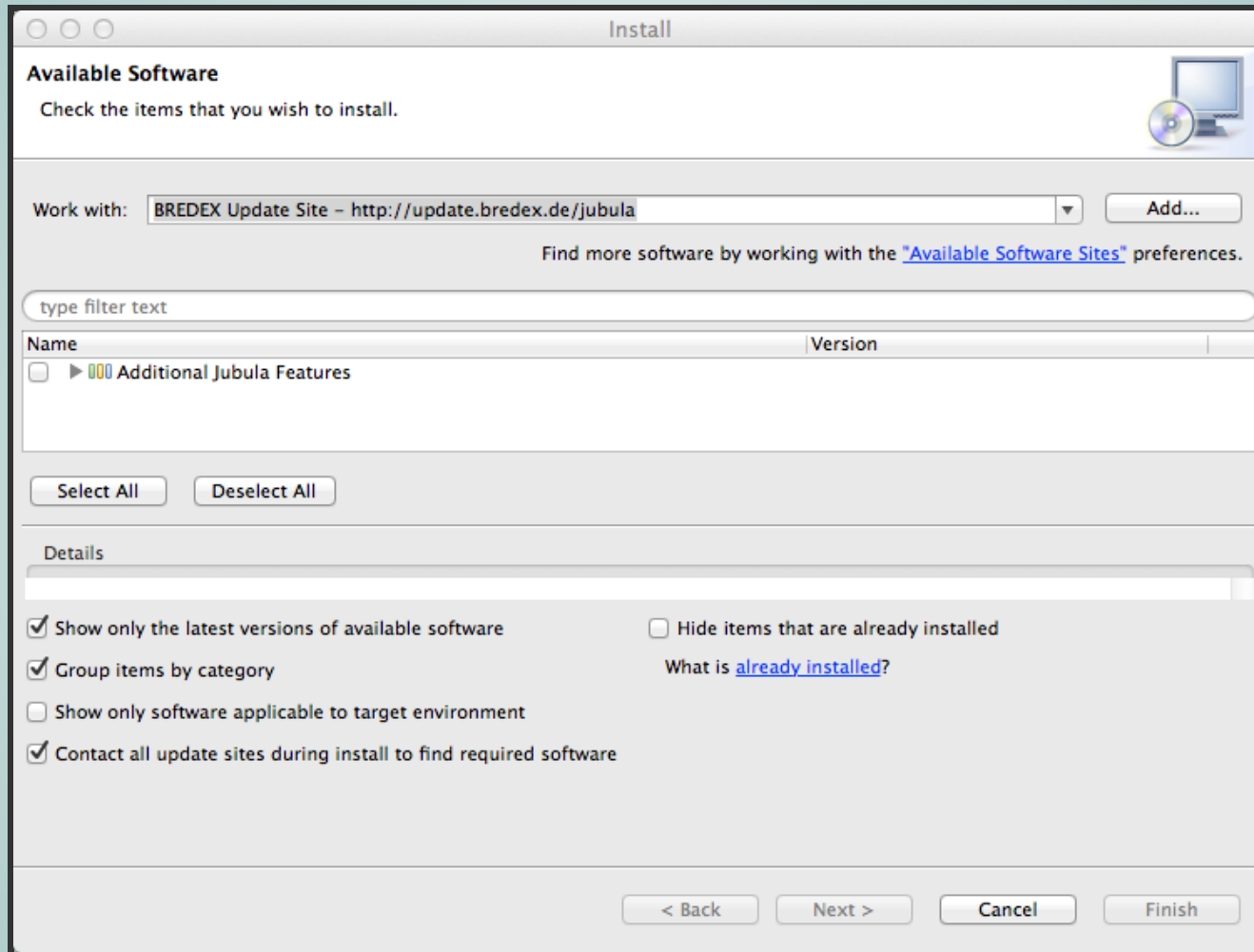
- (1) Use existing extensions
- (2) Write your own extensions
- (3) Build your own Jubula

# EXTENDING JUBULA



- (1) Use existing extensions
- (2) Write your own extensions
- (3) Build your own Jubula

# (1) USE EXISTING EXTENSIONS



## **(2) WRITE EXTENSIONS**

Setup IDE with Jubula as target platform

Write your own bundle / fragment

Make use of extension points

Deploy into RCP application

# (3) BUILDING JUBULA

Clone **git repository**

**Setup IDE** & modify sources directly

```
[~/git/org.eclipse.jubula.core] $ ant  
Buildfile: ~/git/org.eclipse.jubula.core/build.xml  
  
cleanBuild:  
<..>  
BUILD SUCCESSFUL  
Total time: 4 minutes 1 second
```

build on commandline via ant / maven / tycho

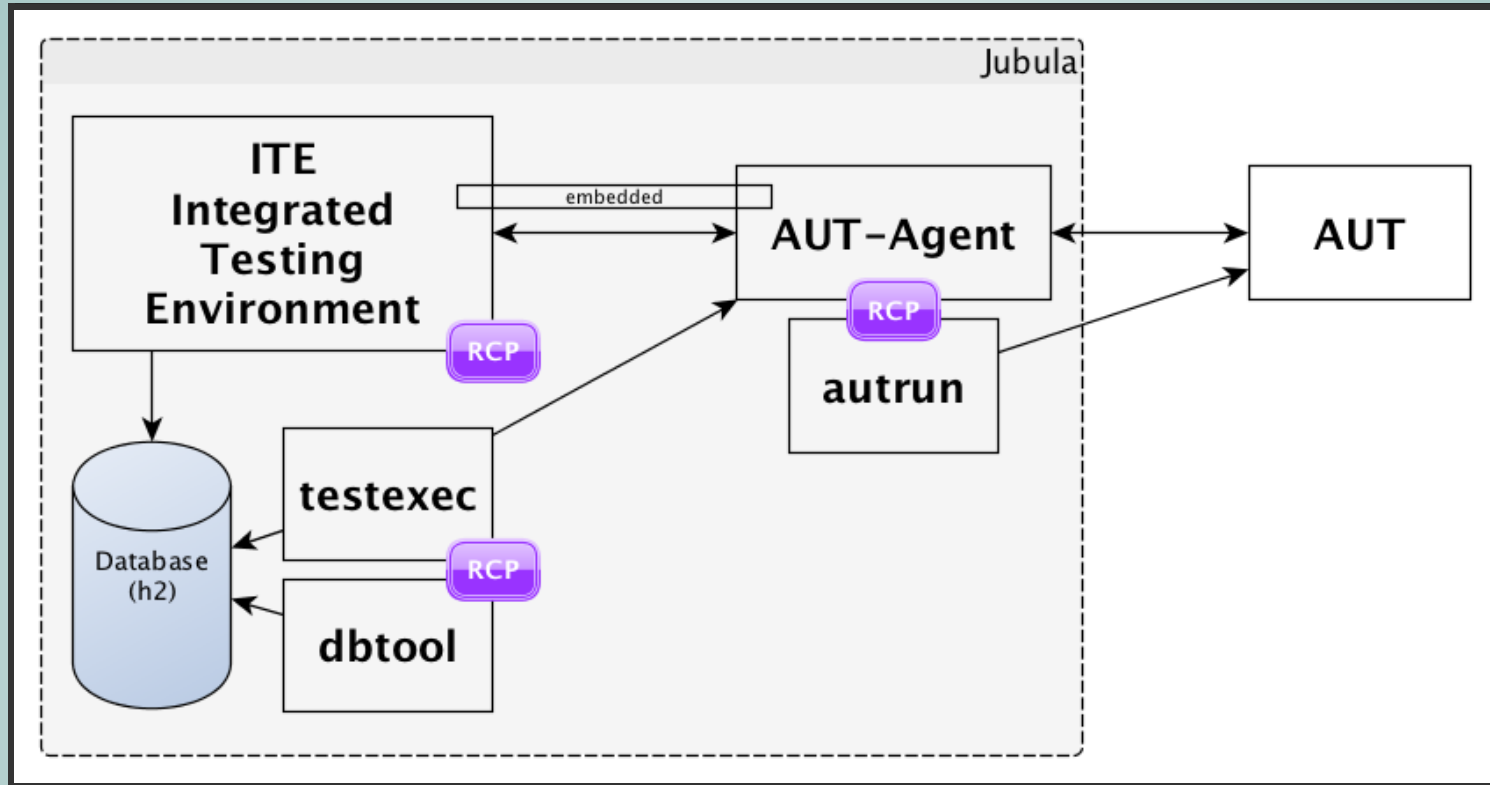
update from local p2-repo [org.eclipse.jubula.site/target/repository/](http://org.eclipse.jubula.site/target/repository/)



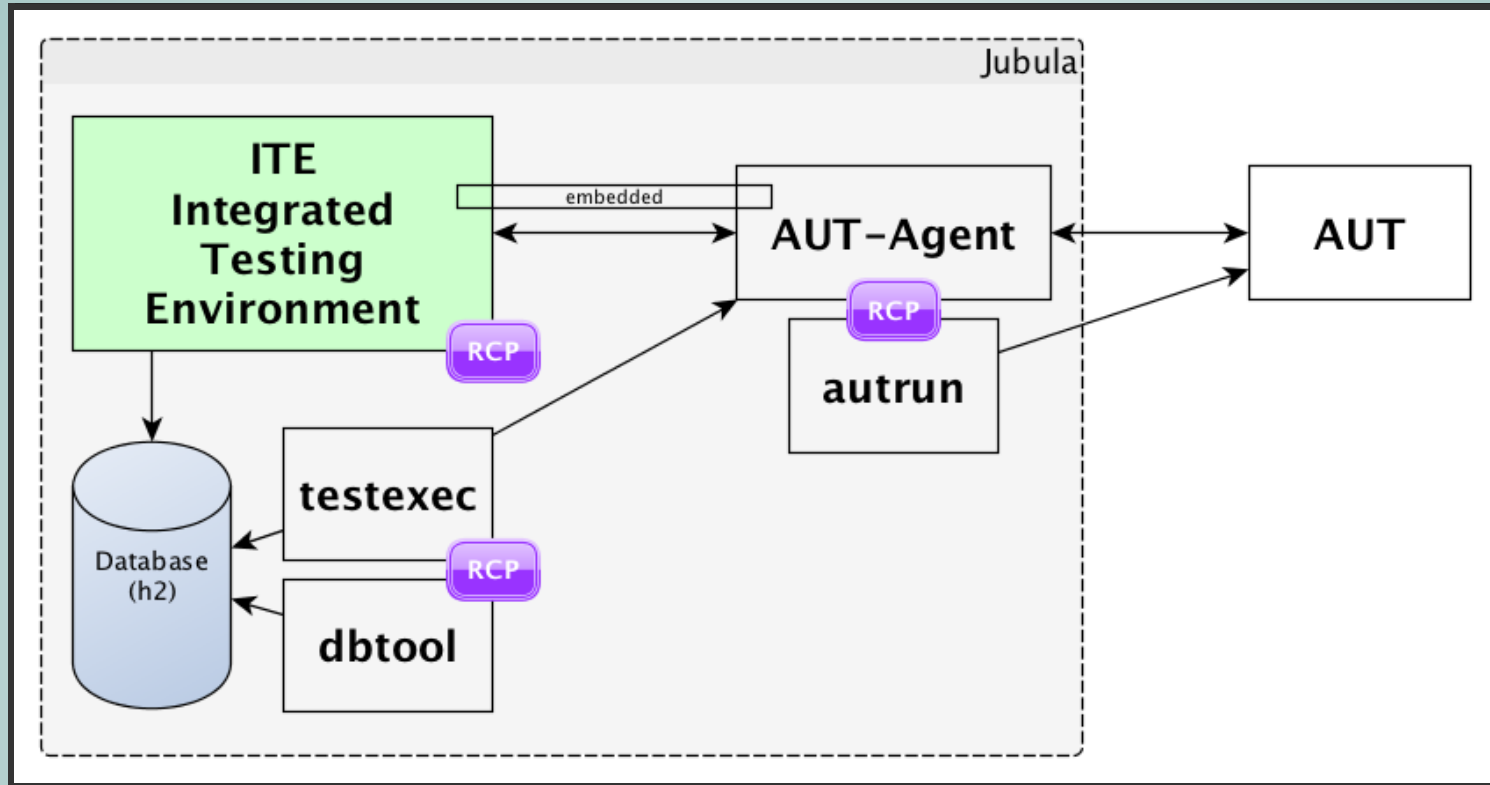




# EXTENDING



# EXTENDING THE ITE



# EXTENDING THE ITE

Write custom **test data functions**

Write your own **test reports**

Write your own **test style rules**

Add your own **unbound\_modules\_\***

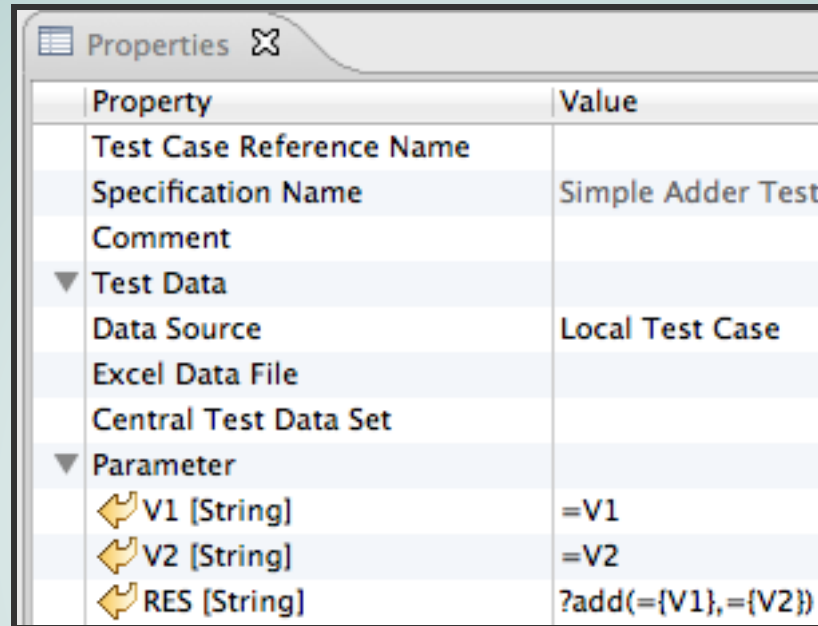
Support more **keyboard layouts**

Support more **ALM repositories**

Support other **databases**

...

# TEST DATA FUNCTIONS



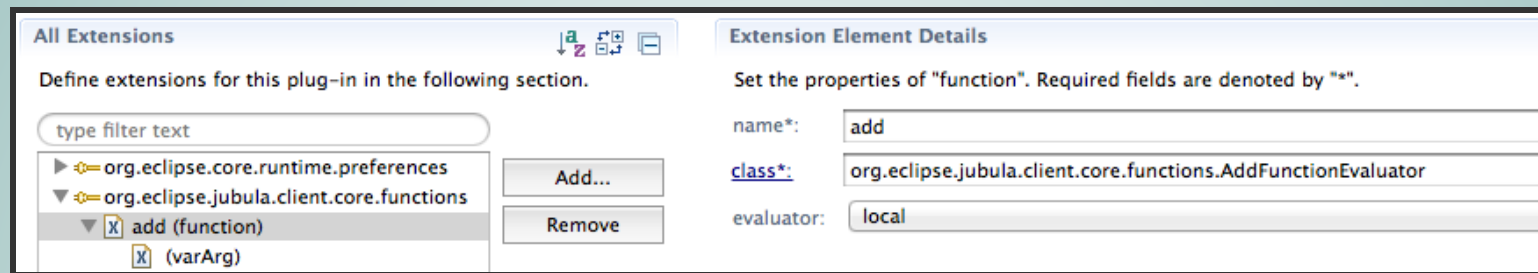
The image shows a 'Properties' window with a table of test case configuration. The table has two columns: 'Property' and 'Value'. The properties include 'Test Case Reference Name', 'Specification Name' (set to 'Simple Adder Test'), 'Comment', 'Test Data' (expanded), 'Data Source' (set to 'Local Test Case'), 'Excel Data File', 'Central Test Data Set', 'Parameter' (expanded), and three parameters: 'V1 [String]' (value '=V1'), 'V2 [String]' (value '=V2'), and 'RES [String]' (value '?add(={V1},{V2})').

Property	Value
Test Case Reference Name	
Specification Name	Simple Adder Test
Comment	
▼ Test Data	
Data Source	Local Test Case
Excel Data File	
Central Test Data Set	
▼ Parameter	
⬅ V1 [String]	=V1
⬅ V2 [String]	=V2
⬅ RES [String]	?add(={V1},{V2})

dynamic test data computation during test execution

# TEST DATA FUNCTIONS

Use extension point: org.eclipse.jubula.client.core.functions

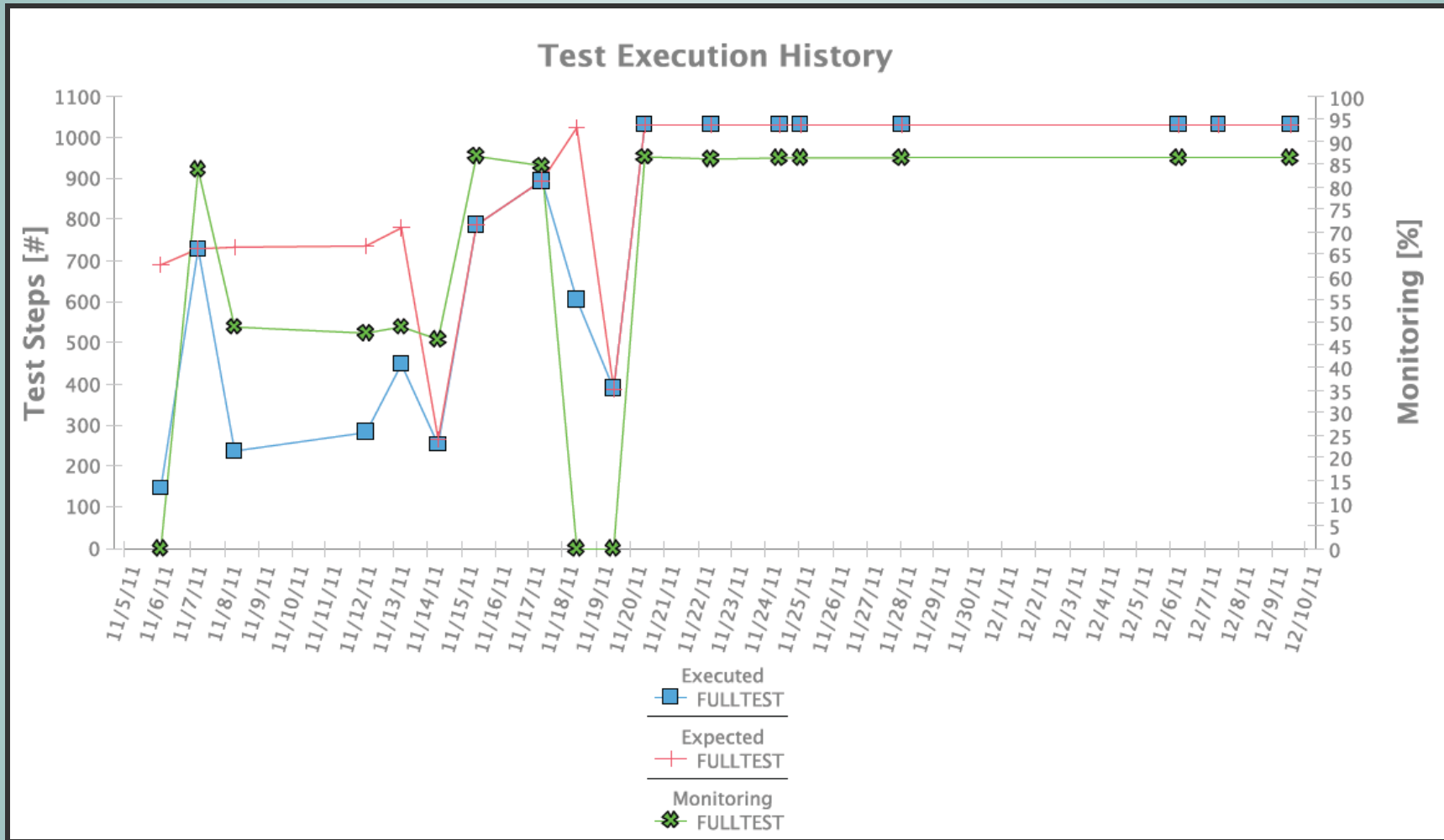


## Implement interface

```
o.e.jubula.client.core.functions.IFunctionEvaluator
```

```
public String evaluate(String[] arguments)  
    throws InvalidDataException;
```

# TEST REPORTS





# TEST REPORTS

use Eclipse BIRT Designer Version 2.6.2

make use of our library: **iteLIB.rptlibrary**

ITE-installDir/jubula/plugins/com.bredexsw.guidancer.

reporting.birt.viewer\_qualifier/reports

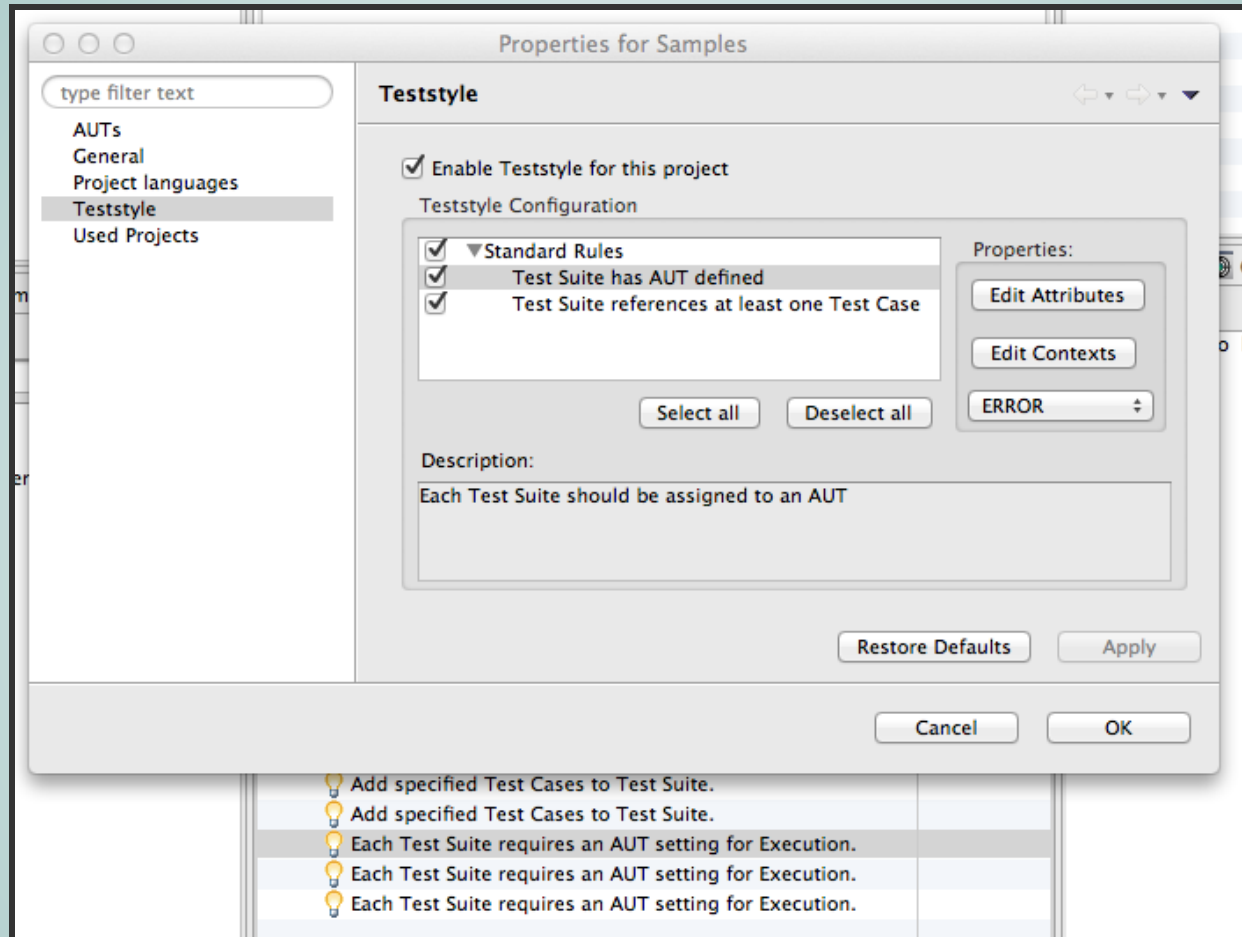
re-use data sources / -sets and reports

deploy to that directory to "install"

recommended: setup central **BIRT viewer instance**

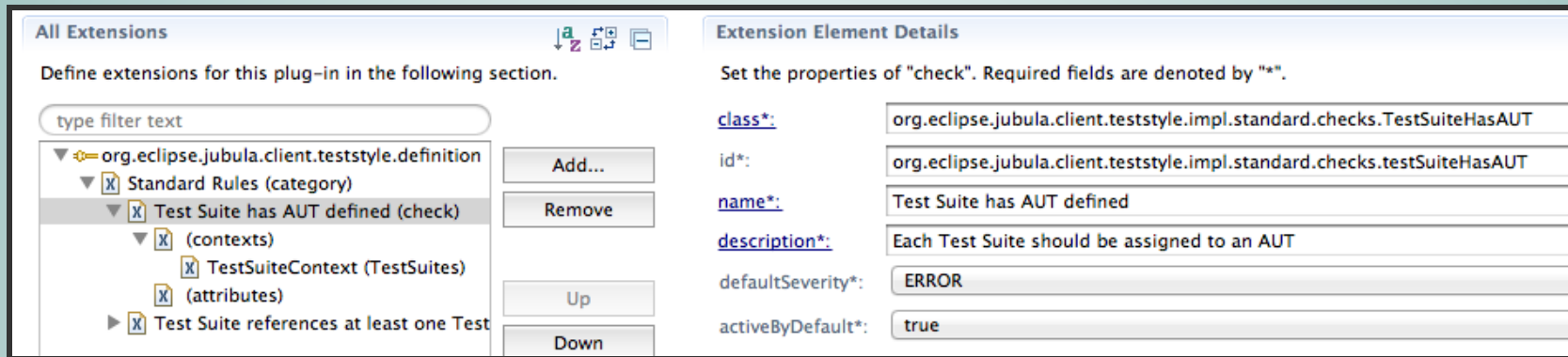


# TEST STYLE RULES



# TEST STYLE RULES

Use extension point: org.eclipse.jubula.client.teststyle.definition

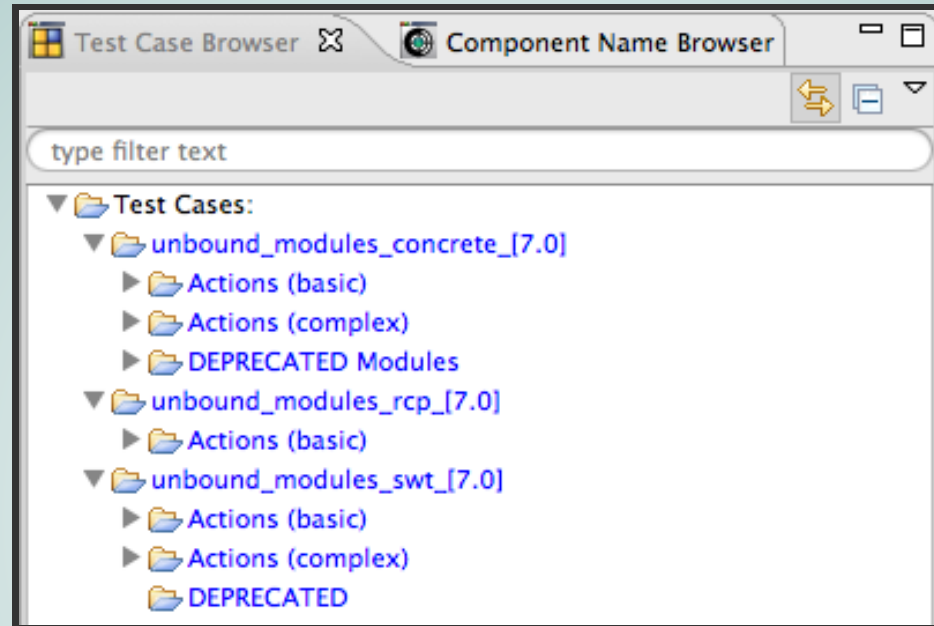


Derive from class:

```
o.e.jubula.client.teststyle.checks.BaseCheck
```

```
// implement  
public boolean hasError(Object obj);
```

# UNBOUND\_MODULES\_\*



pre-defined test case library

# UNBOUND\_MODULES\_\*

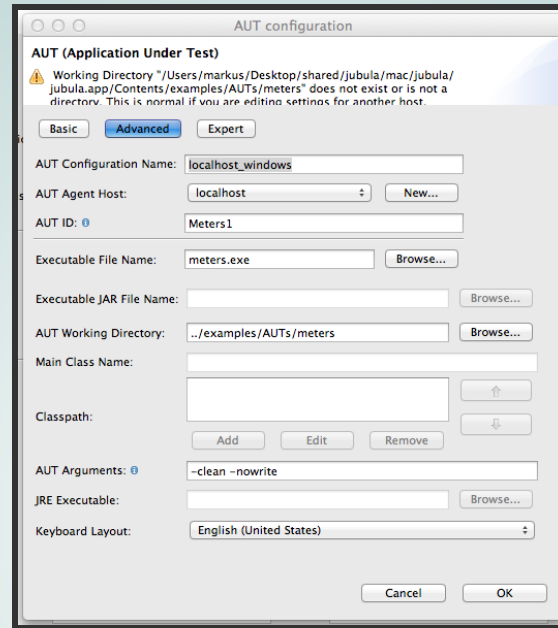
write a fragment for bundle: `org.eclipse.jubula.client.ui.rcp`

provide a file beginning with "unbound\_modules\_"

located under: `resources/library`

--> auto-import on new database scheme

# KEYBOARD LAYOUTS



RCP AUTs require keyboard mapping files  
**de\_DE** & **en\_US** are already pre-defined

# KEYBOARD LAYOUTS

write a fragment for bundle: **org.eclipse.jubula.client.core**

provide a properties file named e.g. "de\_CH.properties"

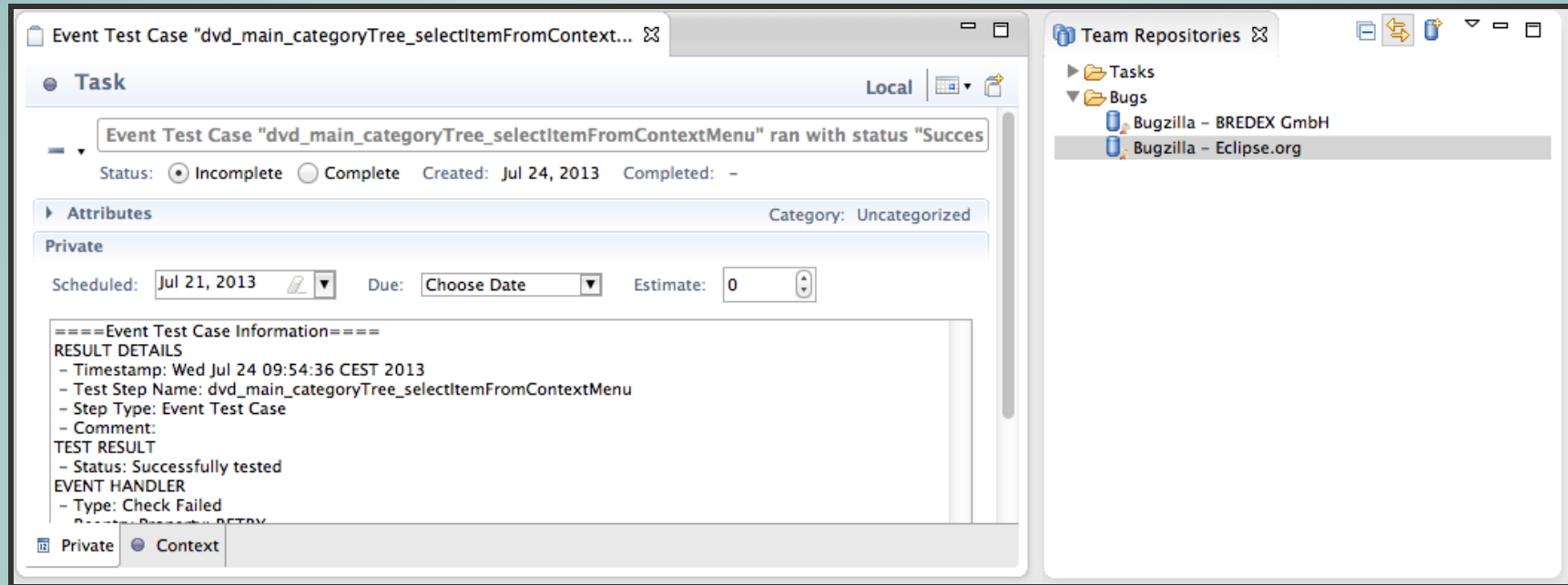
located under: resources/keyboard\_mapping

provide special character mapping

```
...  
"=shift+2  
§=shift+3  
$=shift+4  
%=shift+5  
&=shift+6  
...
```



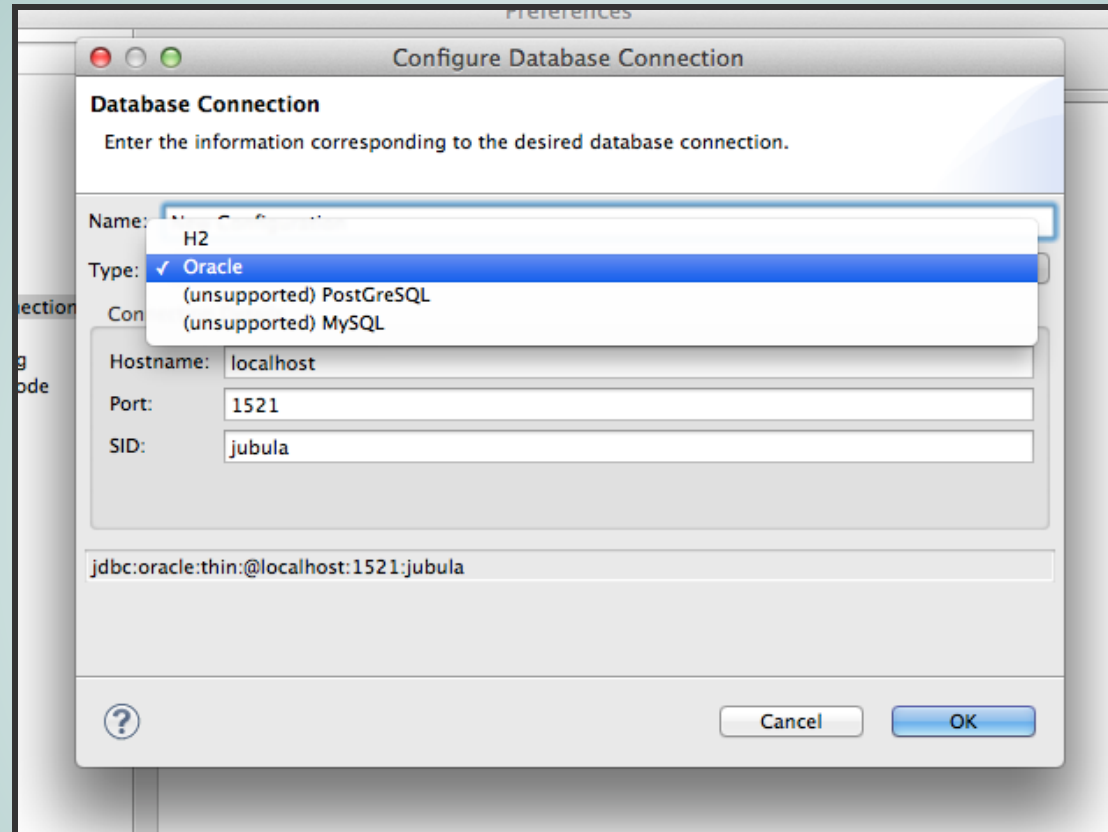
# ALM REPOSITORIES



built-in support for **trac**, **JIRA** & **Bugzilla**  
based on **Mylyn** connectors



# DATABASE TYPES



# DATABASE TYPES

JPA / EclipseLink used for database persistence

potentially much more databases compatible

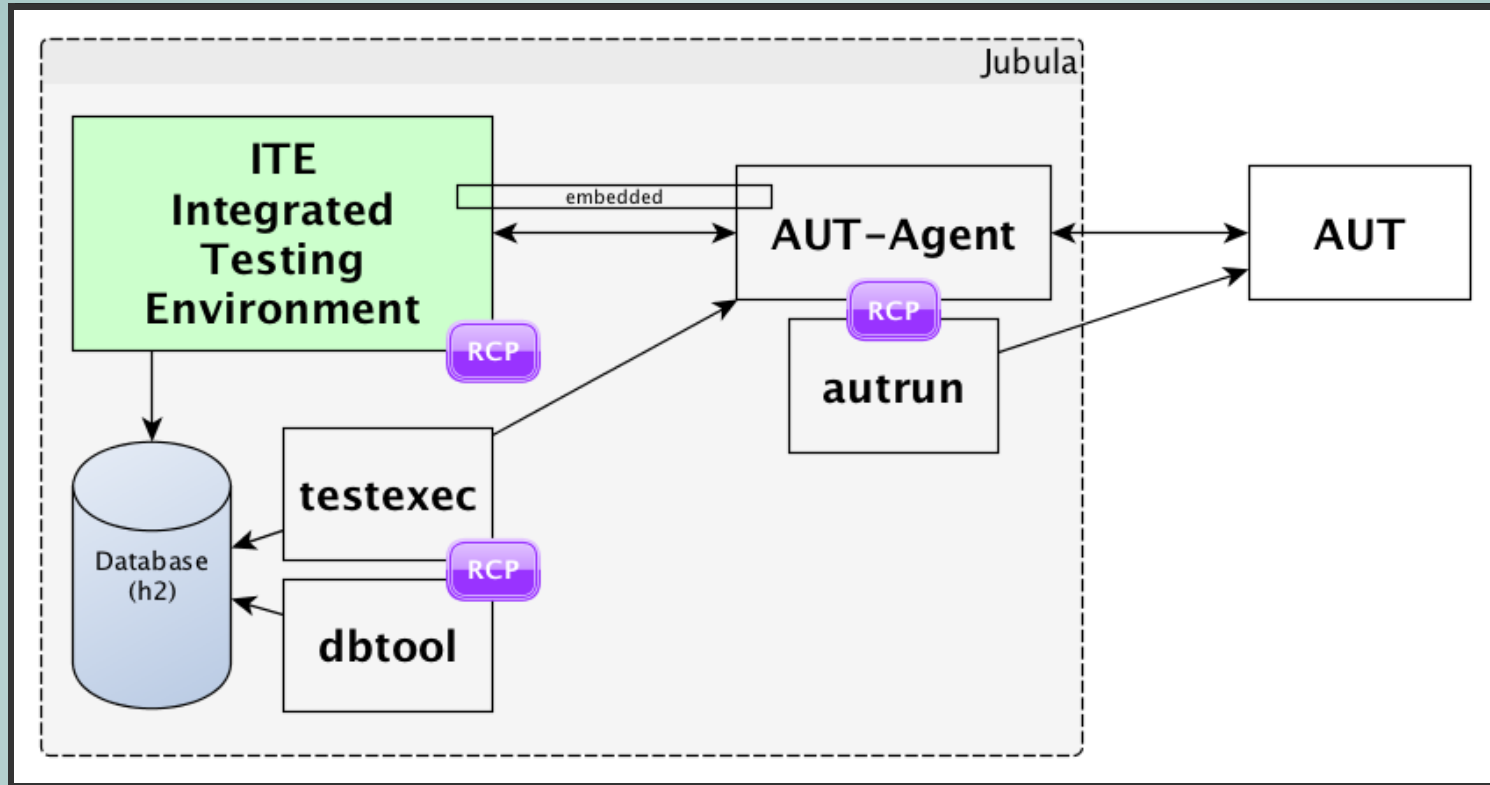
write your own: `o.e.jubula.client.core.preferences.database.`

`AbstractHostBasedConnectionInfo`

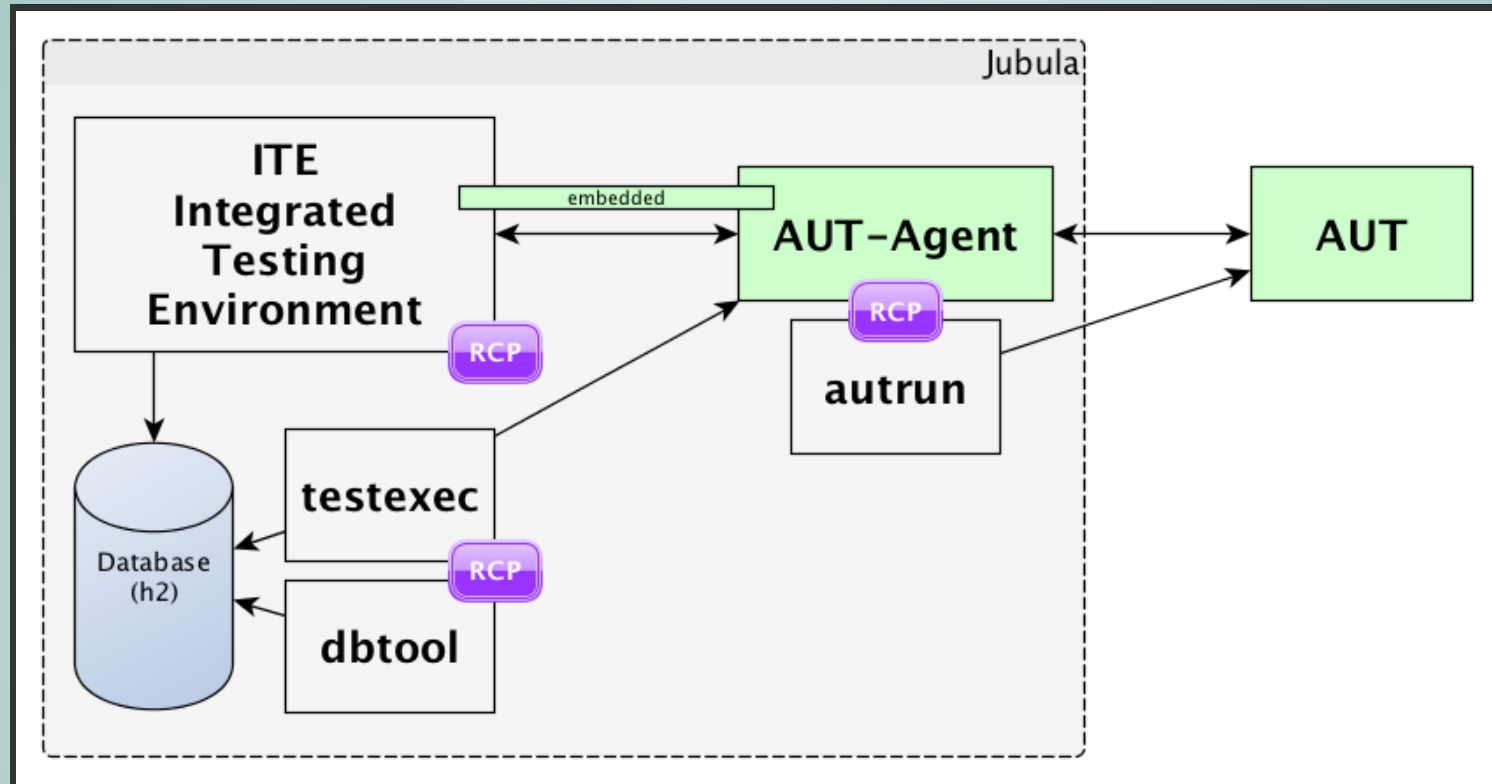
provide JDBC driver

build Jubula on your own

# EXTENDING THE ITE



# EXTENDING THE AUT-AGENT, RC & AUT



# EXTENDING THE AUT-AGENT, RC & AUT

Improving UI-widget recognition in Swing, SWT, HTML, iOS

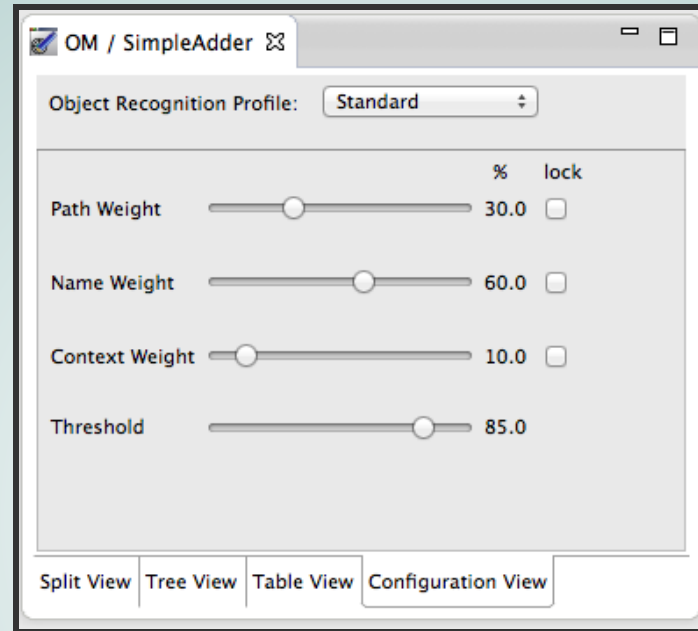
Improving GEF testing by enhancing the identifier

Adding support for custom monitoring agents

Supporting your own components, actions and toolkits

...

# IMPROVING UI-WIDGET RECOGNITION



set unique IDs for UI widgets in your AUT



# IMPROVING UI-WIDGET RECOGNITION

has to be implemented within the AUT itself

**Swing:** setName(ID)

**SWT / RCP:** setData("TEST\_COMP\_NAME", ID)

**HTML:** set / use id attribute id="ID"

**iOS:** UIAccessibilityIdentification Protocol

```
@property(...) NSString *accessibilityIdentifier
```



# IMPROVING GEF TESTING



within FigureCanvas regexp based tree-path is used  
by default no real semantic (type + index)

# IMPROVING GEF TESTING

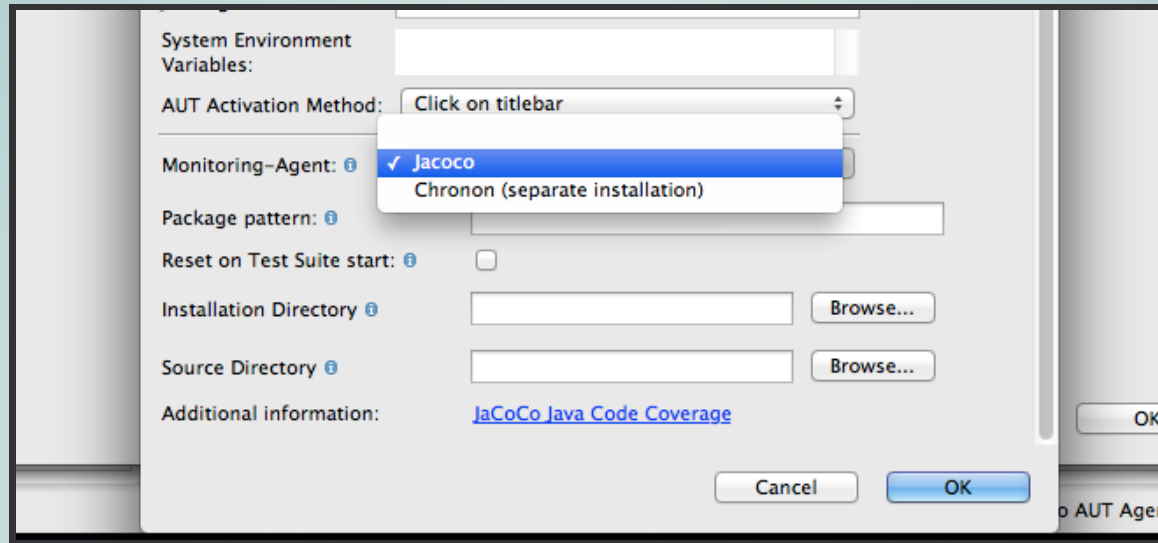
provide adaption to:

`o.e.jubula.rc.rcp.e3.gef.identifier.IEditPartIdentifier`

make use of extension point: `org.eclipse.core.runtime.adapters`

**talk** at ECE 2012

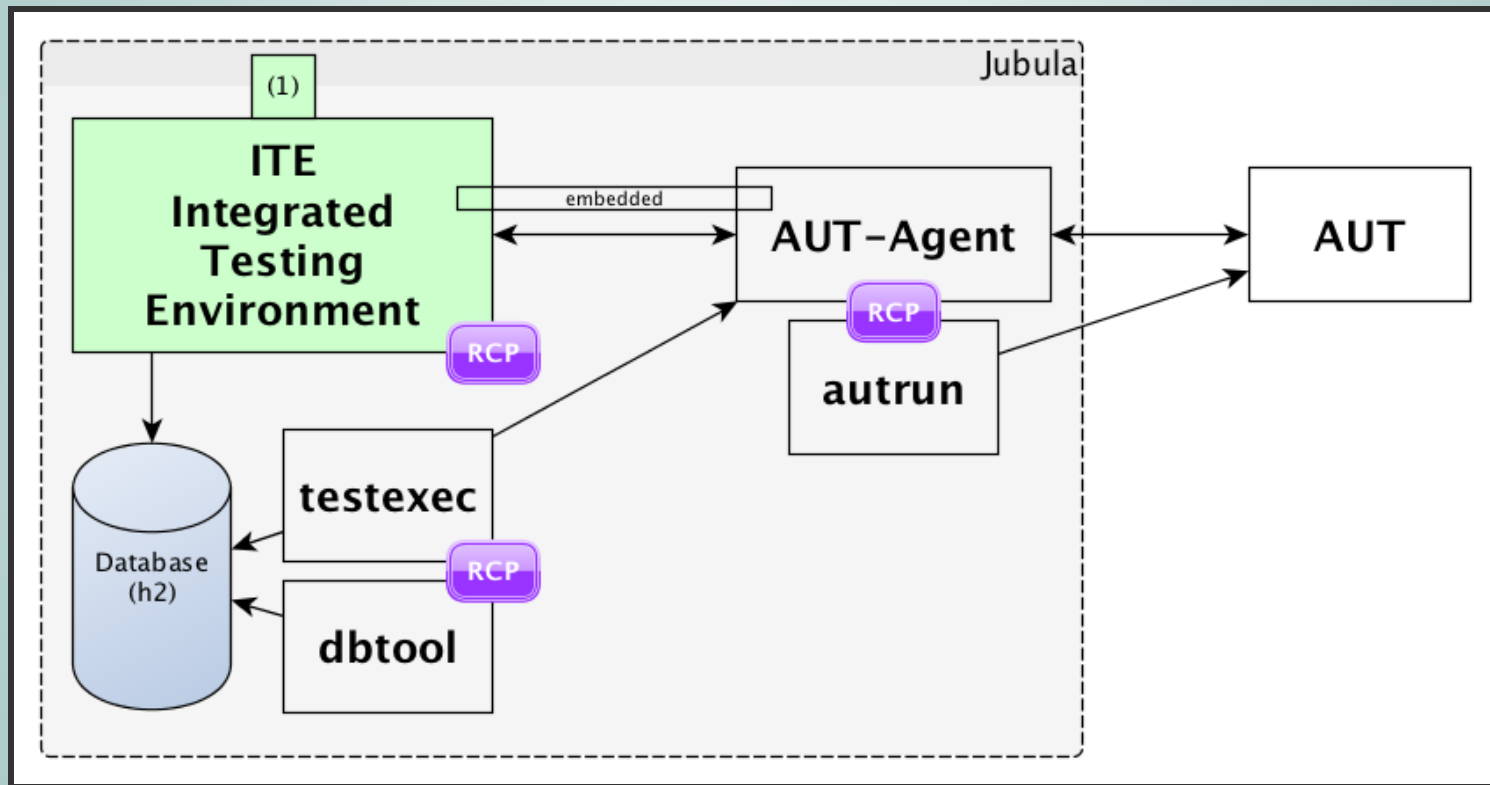
# CUSTOM MONITORING AGENTS



built-in support for **JaCoCo**

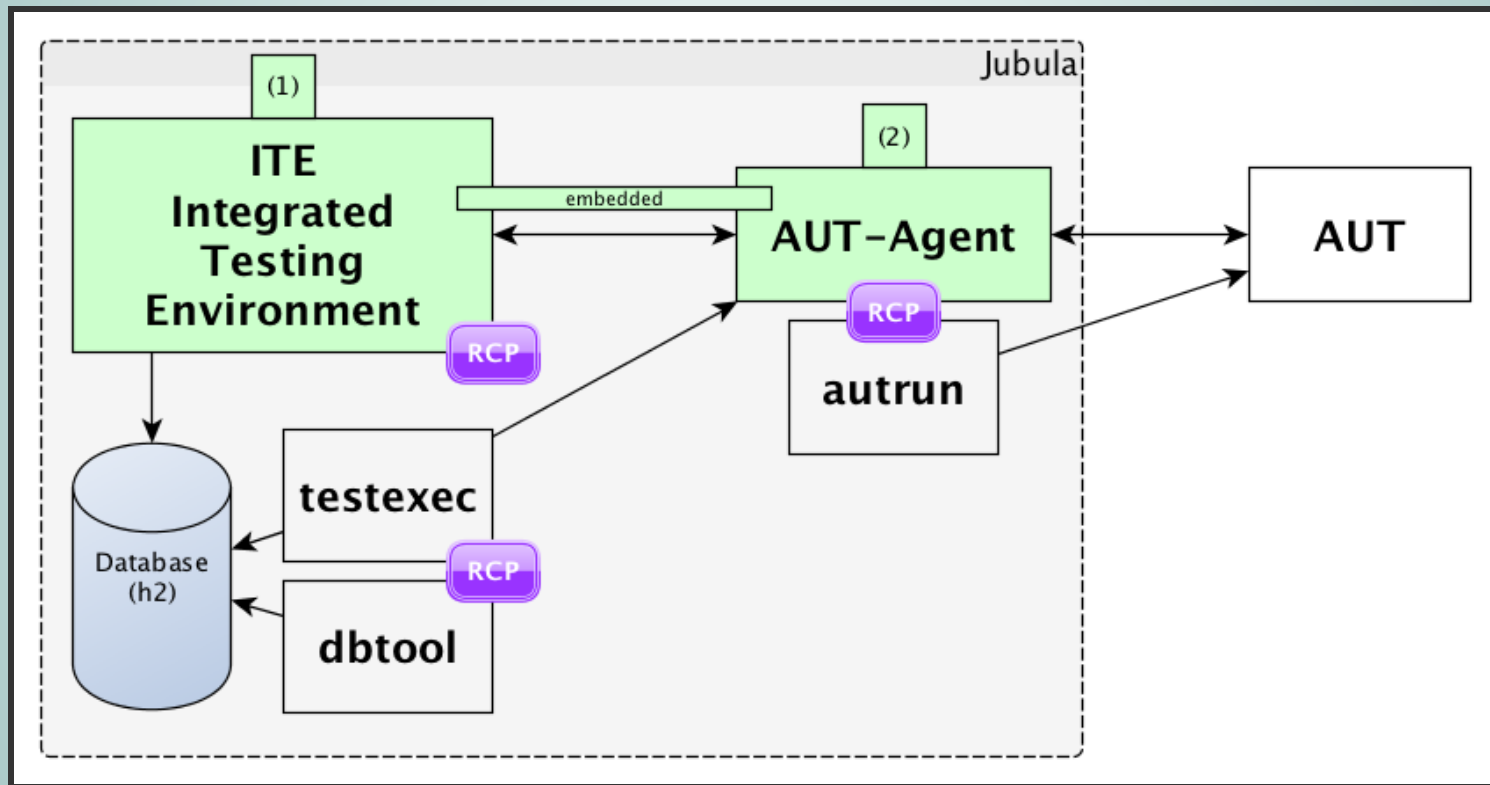
# CUSTOM MONITORING AGENTS

(1) write extension for ITE: `o.e.jubula.toolkit.common.monitoring`  
provide AUT-configuration relevant information



# CUSTOM MONITORING AGENTS

(2) write extension for AUT-Agent: `o.e.jubula.autagent.monitor`  
provide technical realization



# SUPPORTING CUSTOM COMPONENTS

(sometimes) necessary to extend / modify / rewrite UI widgets; e.g.



Many ways of extending – support for:

- (1) custom actions for existing components
- (2) custom components for existing toolkit
- (3) custom (not-yet-built-in) toolkits

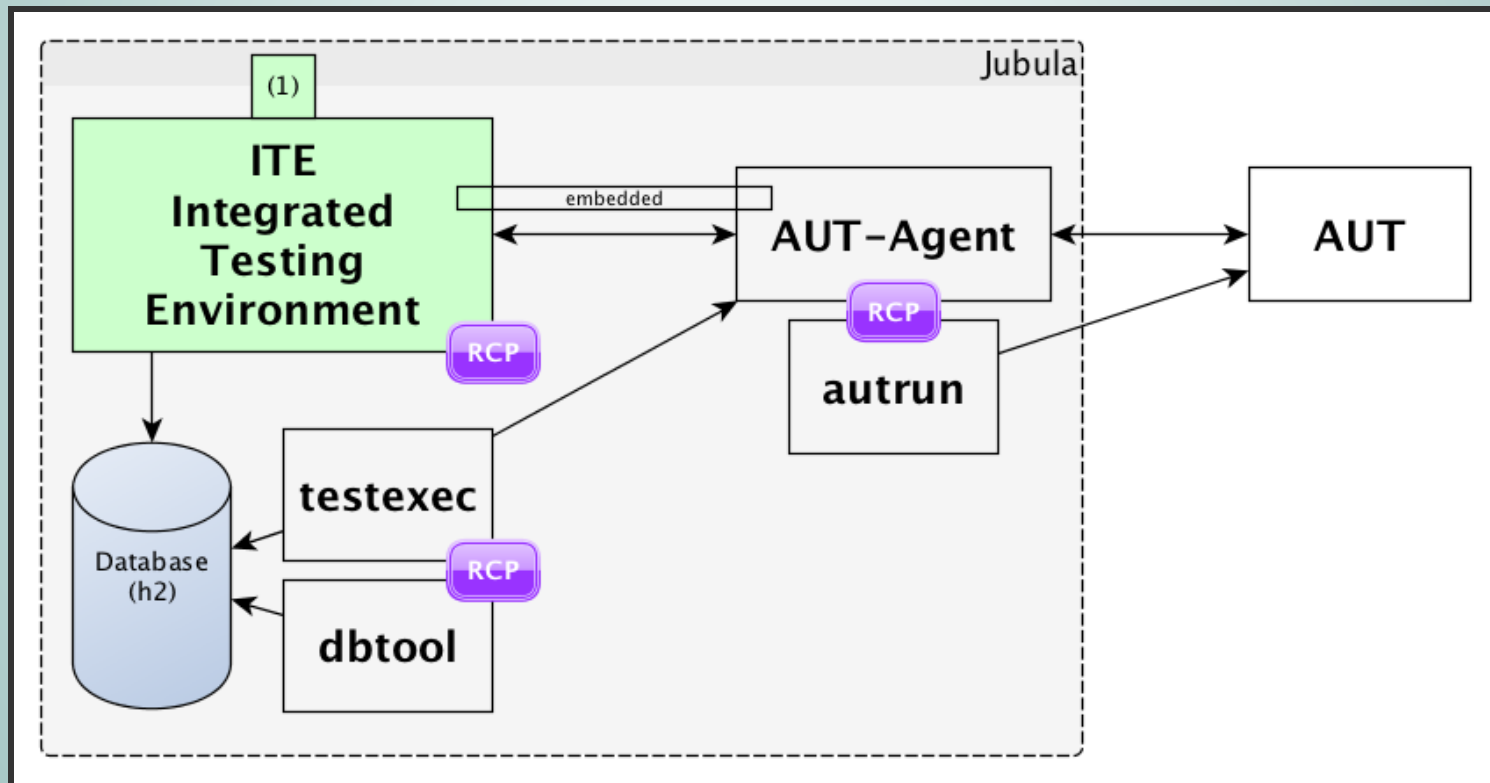


# SUPPORTING CUSTOM COMPONENTS

(1) write extension for ITE:

`o.e.jubula.toolkit.common.toolkitsupport`

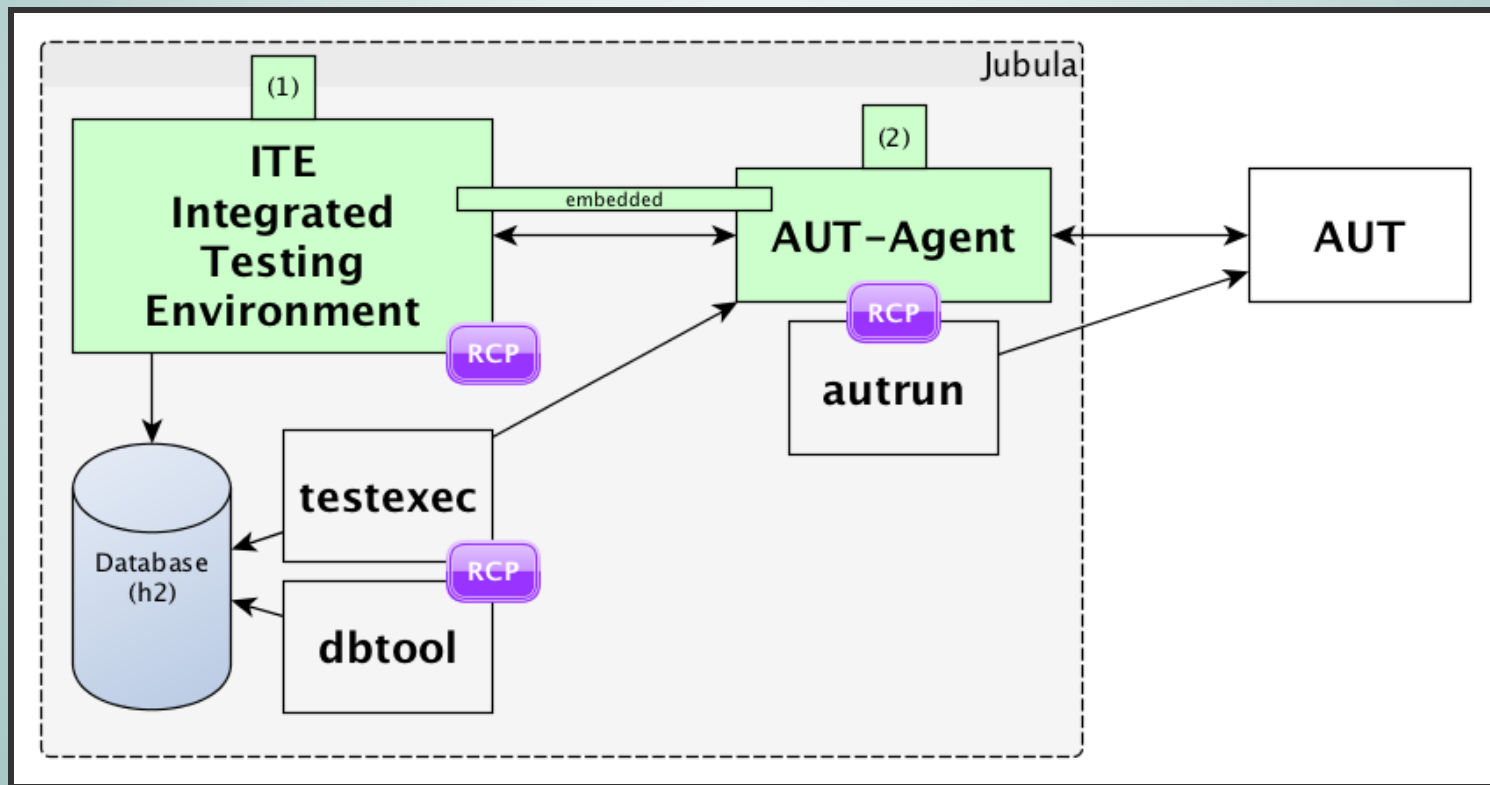
information about type-system and supported CAPs



# SUPPORTING CUSTOM COMPONENTS

(2) provide fragments for RC bundles in AUT-Agent  
provide toolkit dependent technical realization

talk at ECE 2012



# MORE INFORMATION...

Extension API Manual

**Eclipse Jubula**

Jubula **Forum**

Jubula **Developer Mailing List**

**WHEN YOU'RE DONE THINK ABOUT...**  
**CONTRIBUTING!**

**THANK YOU! - Q&A?**


# Give Feedback on the Sessions

1

Sign In: [www.eclipsecon.org](http://www.eclipsecon.org)

2

Select Session Evaluate

Making ALM Work -   
Transform your Application Lifecycle Management to Foster Innovation (presented by HP)  
Ronit [HP]

EVALUATE

3

Vote

+1

0

-1