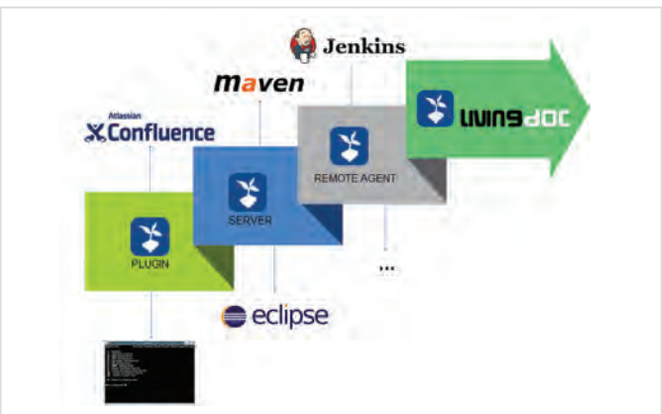
FACTSheet

LivingDoc - Your Open Source Testing Collaboration Solution

LivingDoc is a powerful Open Source tool that supports the implementation of collaboration driven methods like Behavior Driven Development or Specification By Example. Due to its smart integration into Atlassian Confluence it is one of the best tools to evolve a living documentation system, especially if you are already using Atlassian Confluence as your Knowledge Management System. This feature is what sets LivingDoc apart from its competitors. Some of those, while very developer friendly, are not really accepted by experts, like Business Analysts, Experts or Testers, without a strong technical background. Even the reporting features provided by those competitor tools or their extensions/plugins do not measure up to LivingDoc.

Overview LivingDoc



Atlassian Confluence features the LivingDoc plugin. LivingDoc works by executing a Confluence page which calls custom written Fixtures. A Fixture is a bridge between the Confluence page and the System Under Test (SUT) and is written in Java. Whenever a page in Confluence is executed, the Remote Agent receives the pages in HTML format, which are then processed by LivingDoc. The methods of the Fixture are called with the appropriate parameters, which in turn executes a piece of business logic in the SUT software system, and passes the results back to the Confluence page. There it is visually indicated if a test has passed or not.

Creating Executable Specifications

Context Definition (Setup)

Context Definition is used to simplify the creation of a particular state for the System Under Test.

	set up	identification of the set up actions		
header	param 1	param 2	...	param n
element 1	given 1.1	given 1.2	...	given 1.n
...	...	...	...	...
element x	given x.1	given x.2	...	given x.n

The first row of the table indicates the name of the interpreter and the name of the desired state. The second row is called header row and serves to identify the data to be inserted in the System Under Test. Finally, the remaining rows contains the data to be inserted.

Rule Validation (Rule for)

Rule Validation is used to express concrete and measurable business rules. It compares the values returned by the System Under Test against the expected values defined by the Business Expert.

	rule for	identification of the set rule			
header	param 1	param n	...	question 1?	Question n?
example 1	given 1.1	given 1.n	...	expected 1.1	expected 1.n
...	...	...	...	...	...
example x	given x.1	given x.n	...	expected x.1	expected x.n

The first row of the table indicates the set of rules. The next row is called the header row and serves to distinguish the given values and the expected values. Given values serve as inputs to the system, whereas expected values serve as comparison values against values that are actually returned by the system. When a column header ends with special character ? or (), it denotes an expected value. Finally, the remaining rows capture the examples. They are the executable test rows.

Specific Keywords

For the table rule validation you can use two keywords for special cases.These keywords are placed instead of an expected value.

Empty Cells	When a test cell is left blank, LivingDoc only shows the returned value.
Error	When you expect an error, specify it in the cell to test for that particular behavior.

List Validation (List of, Set of, Superset of, Subset of)

List Validations are used to express any kind of groups, lists or sets of values to be compared with the System Under Test. The test result depends on the specific collection selected.

	tabletyp	identification of tabletyp		
header	attribute 1	attribute 2	...	attribute n
element 1	value 1.1	value 1.2	...	value 1.n
...	...	...	...	...
element x	value x.1	value x.2	...	value x.n

As for all other interpreters, the first row of the collection interpreters specifies the name of the interpreter and the name of the collection to be tested. The next row is used to define the name of each attribute related to the members of the collection of data. The following rows are used to specify the set of values to test.

Workflow Validation (Do with / Scenario)

Workflow Validation is used to express interactions with the System Under Test which must be performed in a particular order. This form of specification provides information about the business flow.

	do with	identification of the set rule	
action 1	KeyWord	action 1 description - part 1	parameter 1.1
action 2	KeyWord	action 1 description - part 1	parameter 2.1
...	...	...	...
action x	KeyWord	action 1 description - part 1	parameter x.1

As for all other interpreters, the first row of the DoWithInterpreter specifies the name of the interpreter and the name of the sequence of actions to be tested. What makes the DoWithInterpreter special is that it only has to be defined once for all the sequences of actions expressed in a page. Obviously, the DoWithInterpreter must be defined before any sequence of actions. The following rows are used to express specific actions. The form of each row of a DoWithInterpreter must respect the following rules:

- a row must begin with a part of the action description,
- each parameter must be isolated in a cell,
- each parameter must be separated by parts of the action description.

An action description can be left blank in order to separate two parameters. The DoWithInterpreter provides a minimum of keywords used to define a specific action. The DoWithInterpreter may also be expressed in Bullet List form or Number List form.

Specific Keywords

LivingDoc offers a list of keywords to support the Business Expert. Those keywords are placed at the beginning of an action row.

Accept	Confirm that the action has been executed by the System Under Test.
Check	Verify the specified expected value with the value returned by System Under Test.
Reject	The action should not be performed by the System Under Test (expected errors).
Display	Print the value returned by the System Under Test.

Macros for Confluence Pages

The LivingDoc Confluence plugin places different utility macros at your disposal.

Name and Syntax	Description
Children Macro {livingdoc-children}	Can execute a batch of pages resulting from a page hierarchy.
Historic Macro {livingdoc-historic}	Creates a chart image of the latest historic data of a page execution.The image created provides a clickable area to display the specific execution result.
Import Macro {livingdoc-import: ... }	Allows you to import classes into your executable page without polluting it with undesirable and not meaningful user tables.
Include Macro {livingdoc-include}	Allows you to include pages content into executable document allowing for example, to define a setup and a tear down page and include it in executable pages avoiding duplication.
Info Macro {livingdoc-info}	Allows you to create tables and/or bullet lists that you do not want to be processed as executable specifications within your executable documents.
Labels Macro {livingdoc-labels}	Can execute a batch of documents resulting from a label search.

Writing Fixtures

Generally, Fixtures are glue-codes that connect the functions or methods you want to test and the executable table inside Confluence. For this introduction, we implement functional code instead of calling functions or methods.
You need to open up a Confluence Space, which you have registered as a LivingDoc Space beforehand and create a new page. Now you can start writing your executable specification. To do this, create a new table with at least 3 columns and 3 rows. Since it is a simple example code, a „rule for“ interpreter will suffice. Write it down into the first cell. Next to it type in the name of your Fixture. Your first executable specification can look like this:

rule for	hello world	
first word	second word	concatenate words?
hello	World	helloWorld
living	Doc	livingDoc

It is necessary to add livingdoc-core JAR to the pom.xml of your project. By doing this you can use LivingDoc - Classes.

```
<groupId>info.novatec.testit</groupId>
<artifactId>livingdoc-core</artifactId>
<version>${version}</version>
```

You may have noticed the parameters next to the Annotation FixtureClass and Alias. They are aliases and are directly mapped to the classname.

```
import info.novatec.testit.livingdoc.reflect.
annotation.Alias;

import info.novatec.testit.livingdoc.reflect.
annotation.FixtureClass;

@FixtureClass(„HelloWorld“)
public class HelloWorldFixture {

    @Alias(„first word“)
    public String first;

    @Alias(„second word“)
    public String second;

    @Alias(„concatenate two words“)
    public String concatenateWords(){
        return first + second;
    }
}
```

Specification execution via Confluence

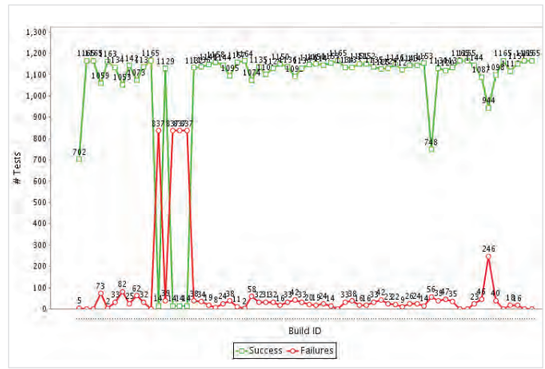
By enabling LivingDoc enabled check box your Confluence page will be activated as a LivingDoc page and the Execute button will be visible.

Reports in Confluence

After a page is executed you will immediately get result reports for it. Following report is a bad case example:

Reports in Jenkins

Your Jenkins records all result statictics after every build. These statictics are visualized in a chart. LivingDoc can be integrated in your Continuous Integration process.



Get in contact with us!

Anis Ben Hamidene
Senior Managing Consultant
NovaTec Consulting GmbH
Head Of Competence Area Agile Quality Engineering

Phone: +49 711 22040-700
Mobile: +49 170 7998 715
Mail: anis.benhamidene@novatec-gmbh.de

LivingDoc is created and developed by NOVATEC

NovaTec Consulting GmbH is an owner-operated, independent German IT consulting firm founded in 1996 supporting customers across various industries in the successful implementation of IT-related projects.

NovaTec Consulting GmbH
Dieselstraße 18/1
D-70771 Leinfelden-Echterdingen
www.novatec-gmbh.de

Twitter
@NT_AQE

Website
www.novatec-gmbh.de/ld

Blog
www.novatec-gmbh.de/aqe-blog

YouTube
www.novatec-gmbh.de/ld-youtube

GitHub
www.novatec-gmbh.de/ld-github

LivingDoc Documentation
ww.novatec-gmbh.de/ld-doc

Trainings
LivingDoc: www.novatec-gmbh.de/ld-training
Specification By Example: www.novatec-gmbh.de/sbe-training