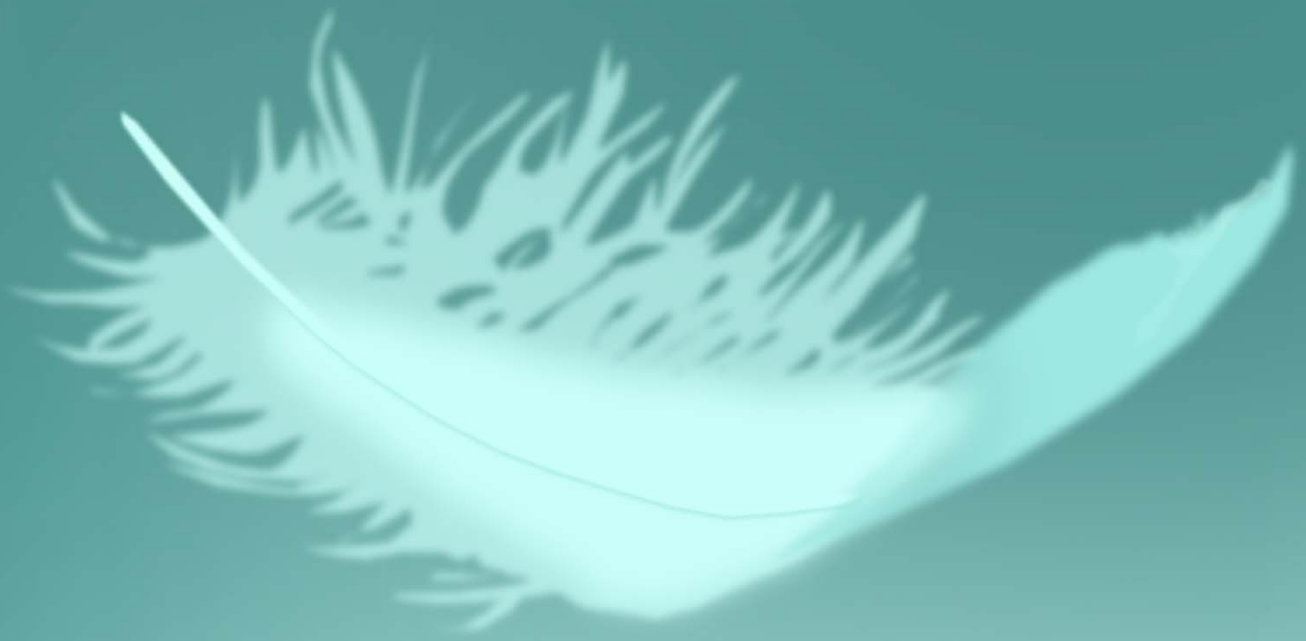
BREDEX
Software-Entwicklung und Beratung



Acceptance testing for Eclipse Applications

Why? What? How?

Alexandra Imrie BREDEX GmbH

 Version 2.3

Agenda

- Observations on test and development process
- The role of acceptance testing
- Methods of acceptance testing
- Introducing and improving acceptance testing

BREDEX GmbH

- **Software developing and consulting**

  Individual customer projects

  Focus on Java since 1995
- **Agile processes**
- **Test enthusiasts**

  Test tool GUI*dancer*
- **Eclipse foundation members since 2006**

Observations

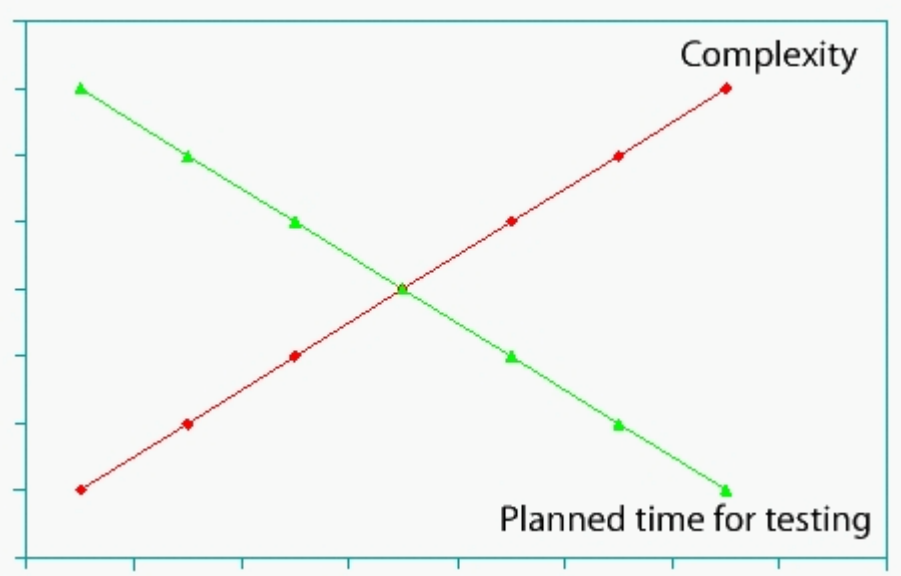
- **Where does acceptance testing fit in?**

Observations

- **Where does acceptance testing fit in?**

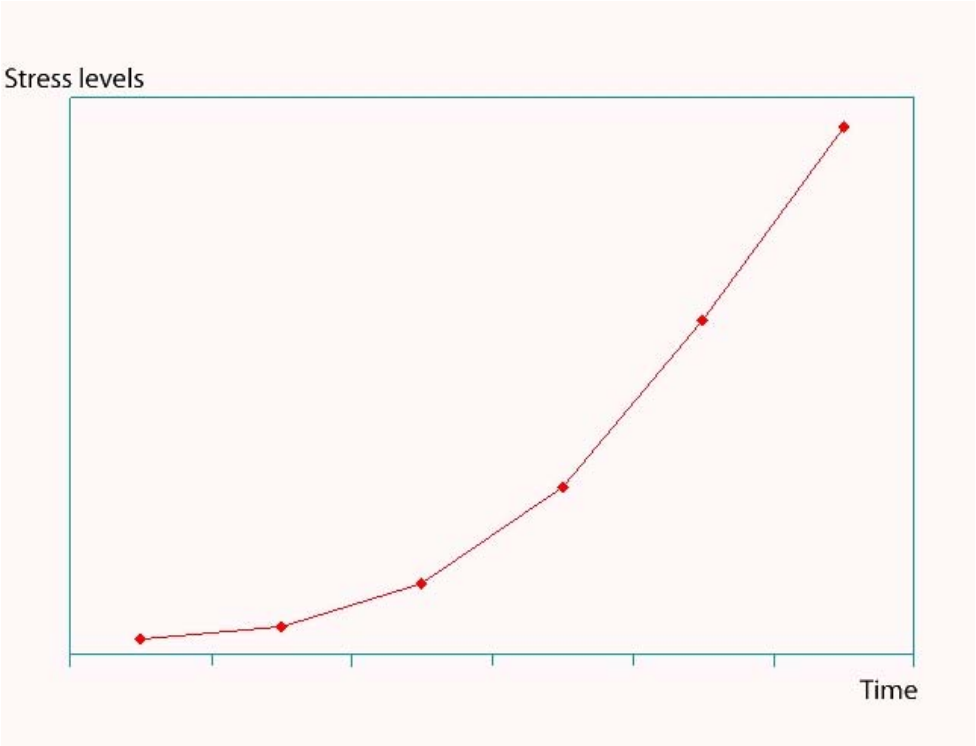
Trends in testing

- **Complexity and test phase**

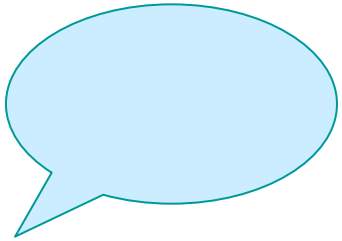


Trends in testing

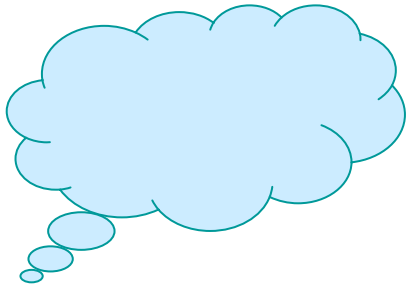
- **Stress towards release**



Responses



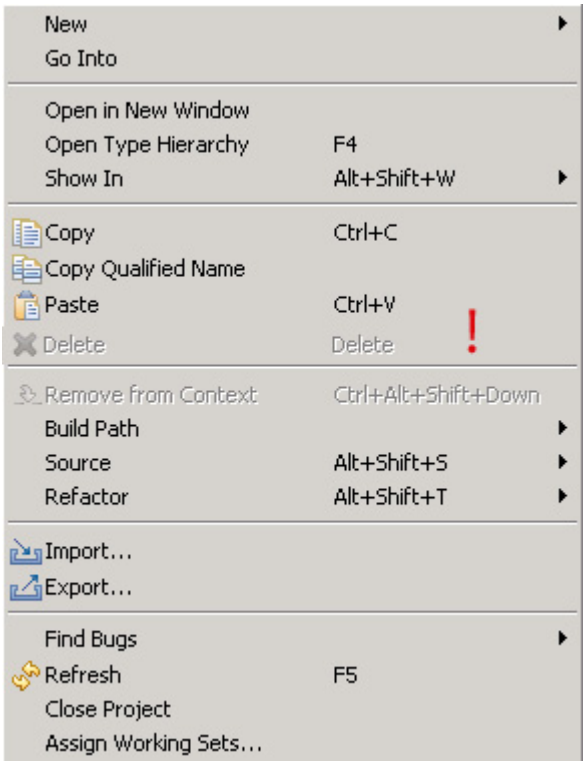
- Where did that come from?
- How come this error wasn‘t noticed?

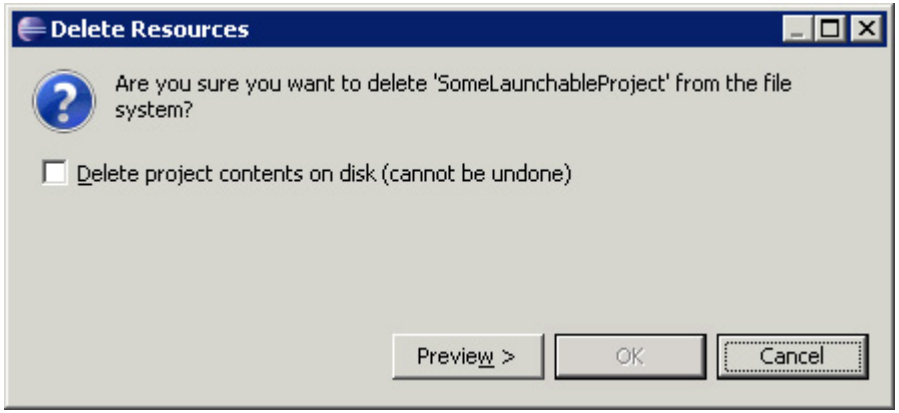


- Desire for quicker feedback on errors
- Move towards test first with JUnit

But JUnit isn't an acceptance test

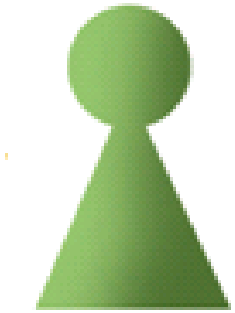
- JUnit test for delete project successful, but…





Paradox

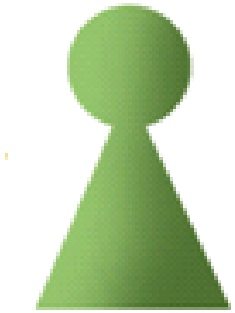
- Who do we develop software for?



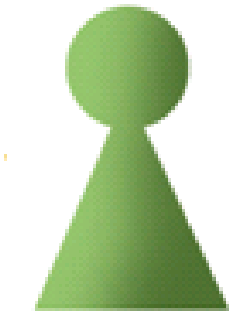
- Whose perspective do we tend to neglect the most?

Paradox

- Who do we develop software for?



- Whose perspective do we tend to neglect the most?



The role of acceptance testing

- Did we do what the customer asked for?
- Has the customer changed his mind?
- Did the customer have implicit assumptions?
- Are there errors or problems in the workflow?

Example

- **Acceptance test to create and delete a Java Project in Eclipse**

  Prerequisite: The project does not already exist

  Open new project dialog

  Choose Java as the project type

  Click Next

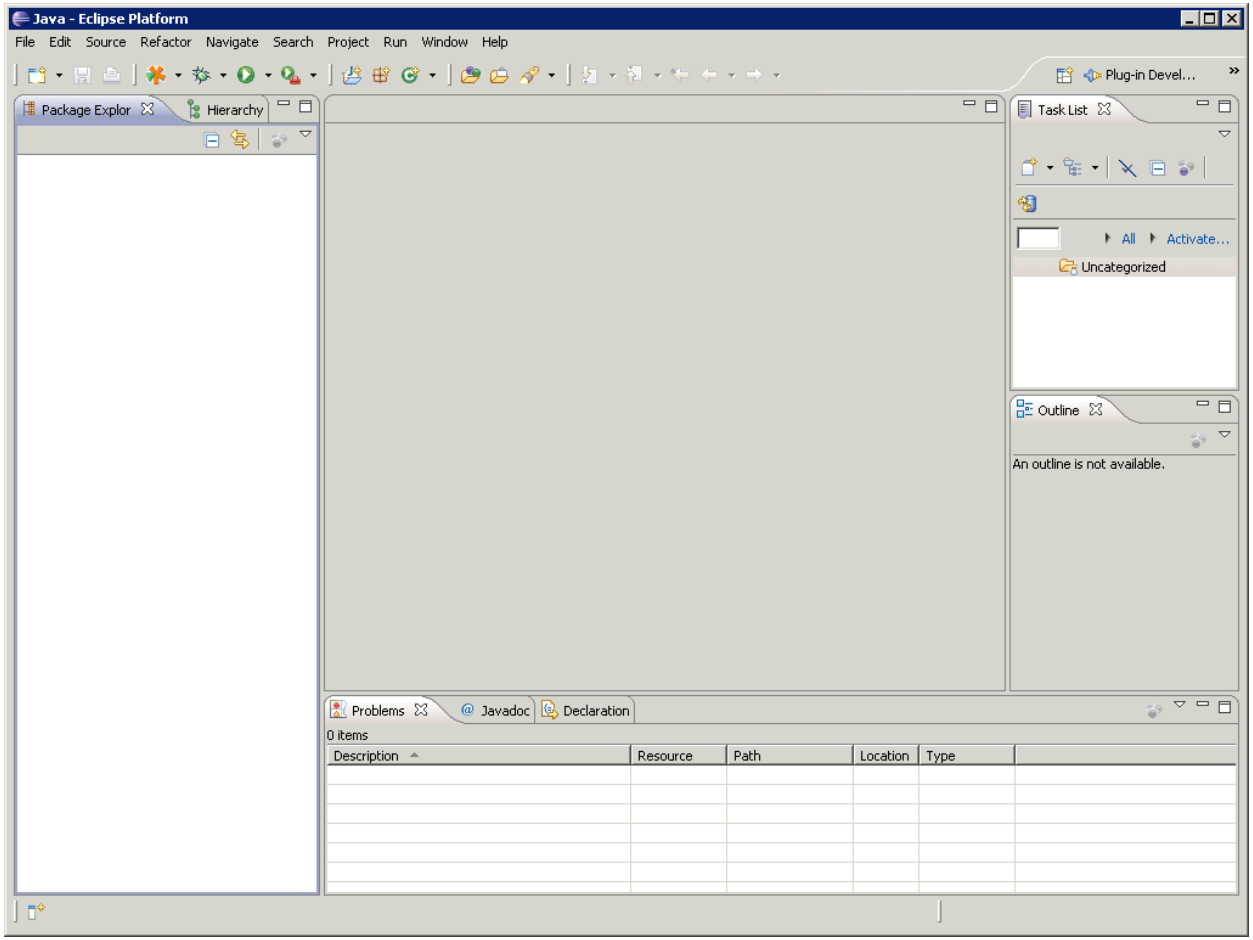
  Enter the project name

  Click Finish

  Check that project was created

  Delete project

Method 1 - Manual



Method 1 - Manual

Positive aspects

Real person performs tests

Negative aspects

Boring

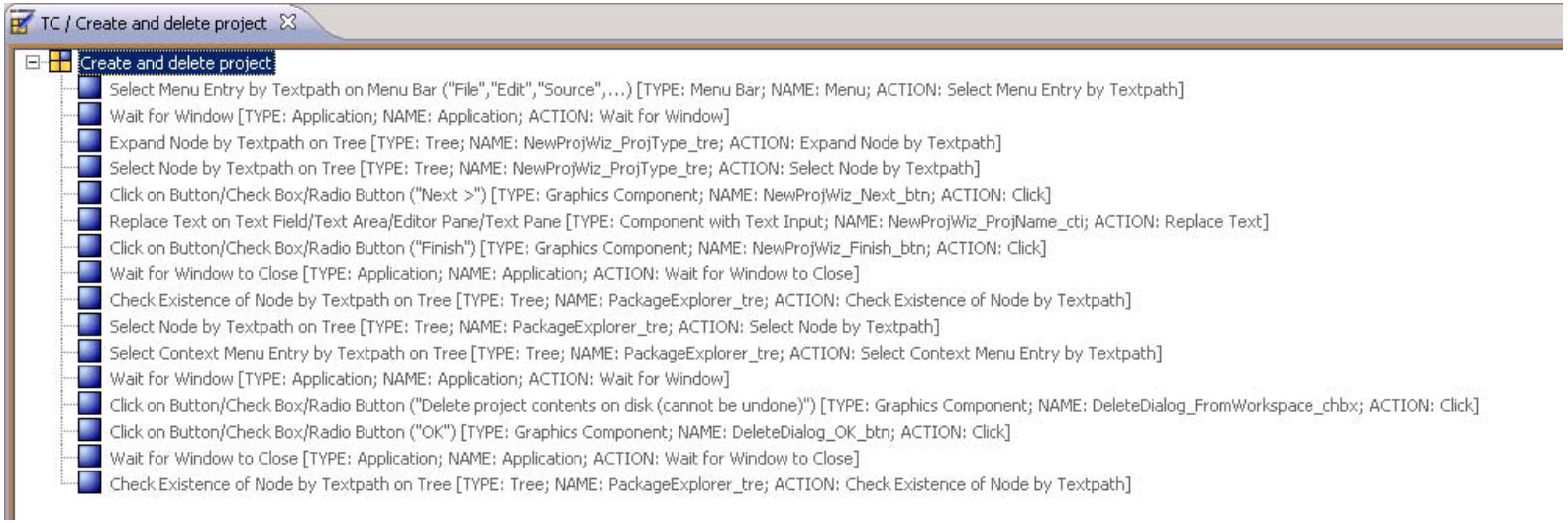
Error prone

Not repeatable

Too much to test

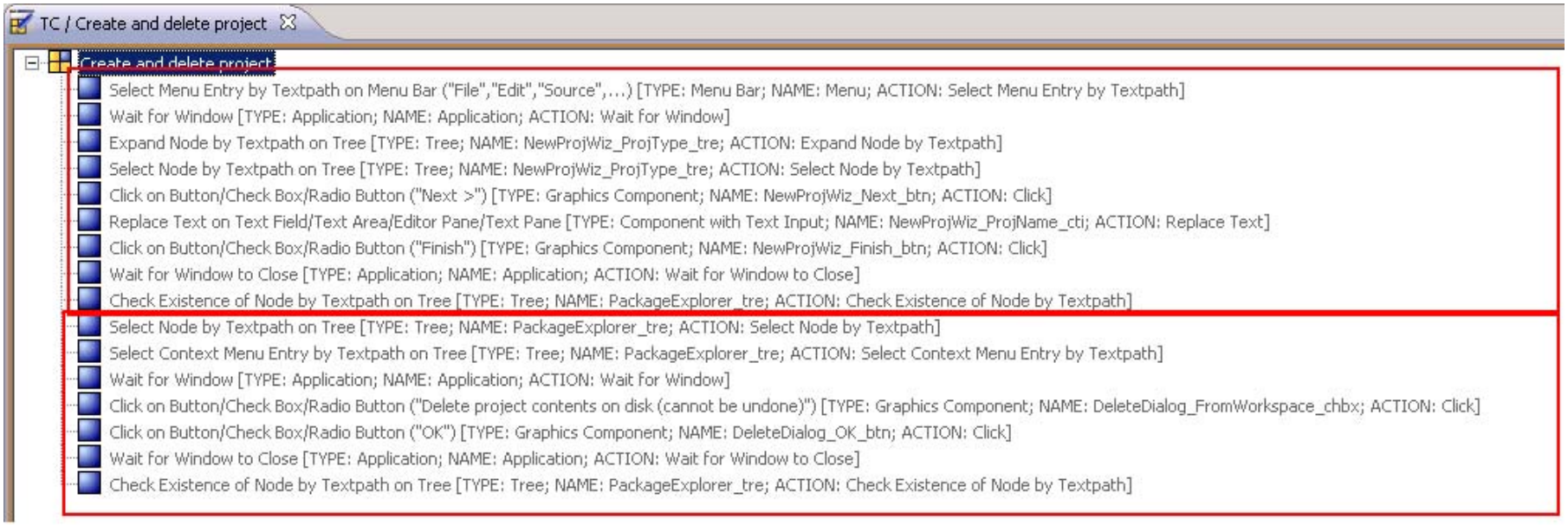
Always at square one

Method 2 – Automated by recording

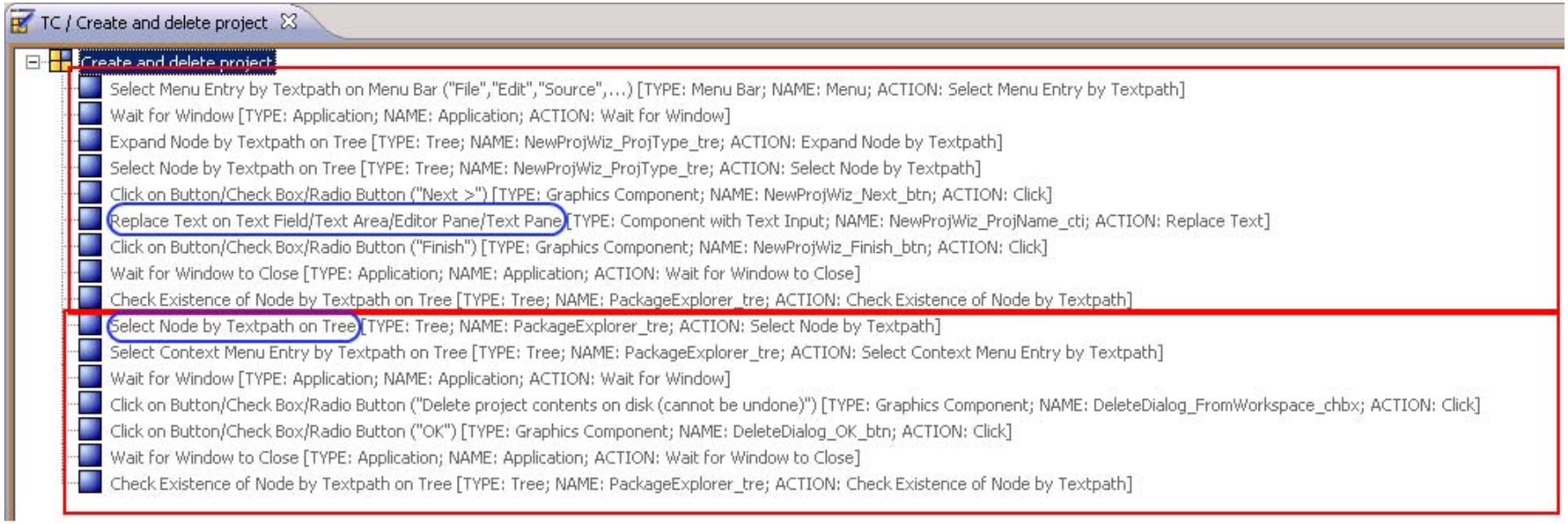



```
if window('Java - Eclipse SDK'):
    select_menu('File>>New...>>Project...')
    if window('New Project'):
        select('org.eclipse.swt.widgets.Tree_1', '/Java/Java Project')
        click('Next')
    select('TextField', 'MyLovelyProject')
    click('Finish')
    close()
    assert('org.eclipse.swt.widgets.Tree_2', 'Text' 'MyLovelyProject', 'true')
    click('org.eclipse.swt.widgets.Tree_1', '/MyLovelyProject')
    select('org.eclipse.swt.widgets.Tree_1', '/MyLovelyProject')
    rightclick('org.eclipse.swt.widgets.Tree_1', '/MyLovelyProject')
    select_menu('Delete...')
    if window('Delete Confirmation'):
        select('org.eclipse.swt.widgets.Button_1', 'true')
        click('OK')
    close()
    assert('org.eclipse.swt.widgets.Tree_2', 'Text' 'MyLovelyProject', 'false')
```

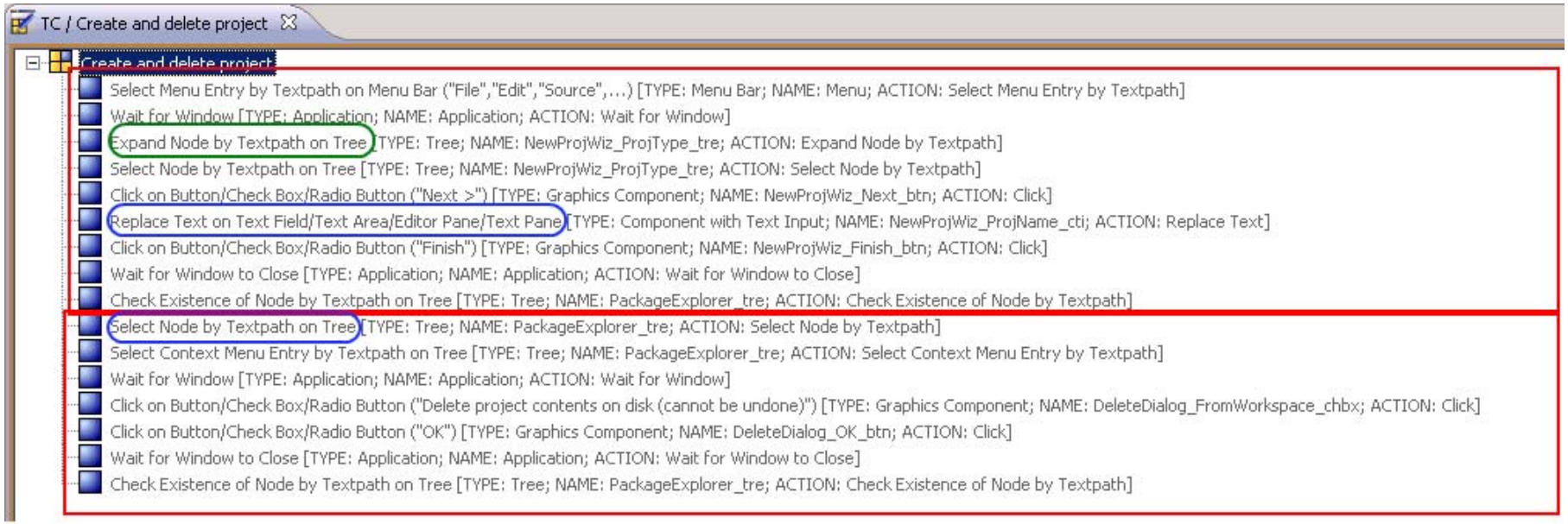
Method 2 – Automated by recording



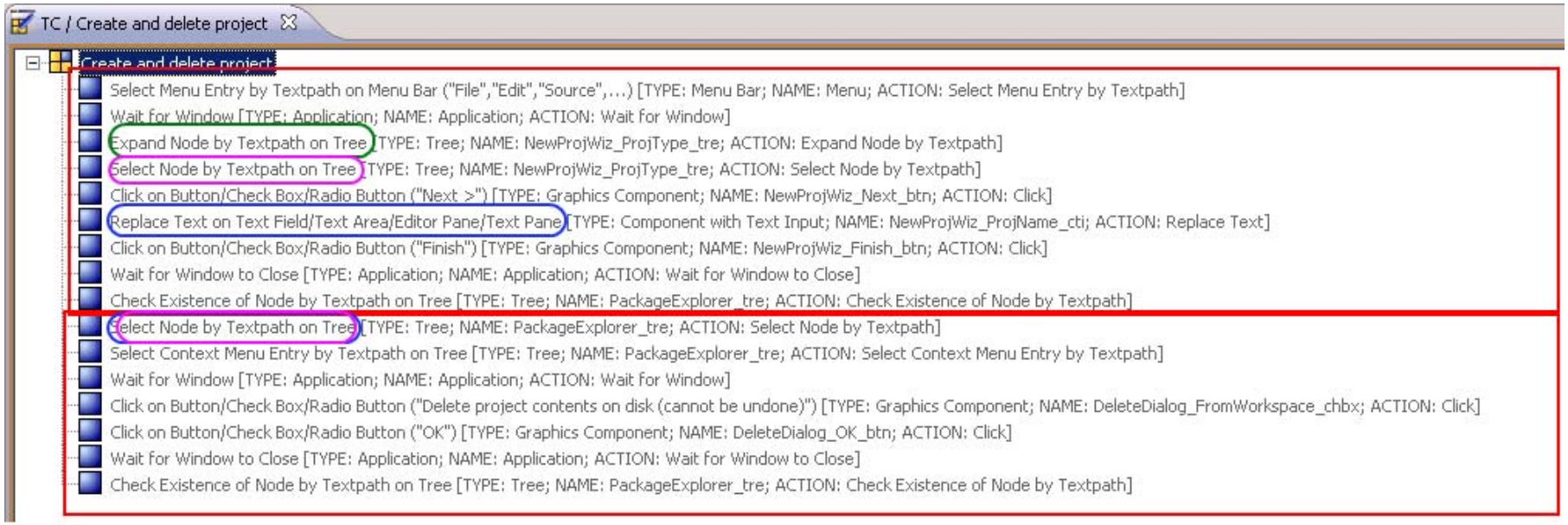
Method 2 – Automated by recording



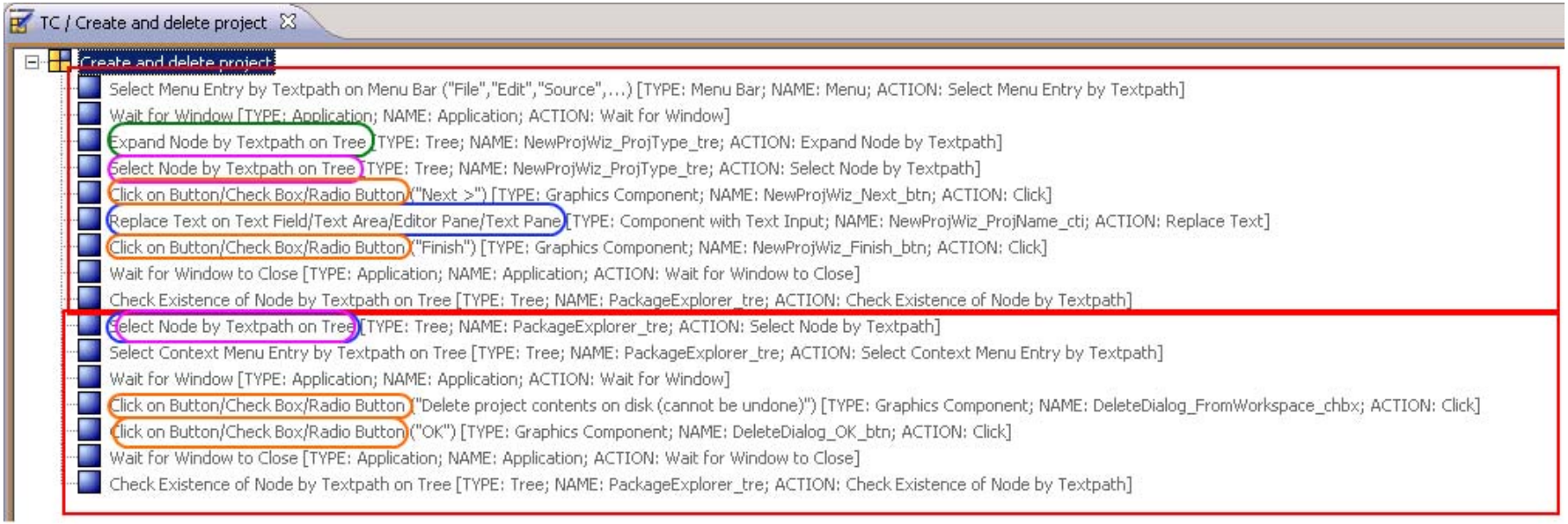
Method 2 – Automated by recording



Method 2 – Automated by recording



Method 2 – Automated by recording



Method 2 – Automated by recording

- **Positive aspects**

Quick start

- **Negative aspects**

Wait for functioning feature

Raw form:

→ unstructured

→ redundancies

→ hard to read

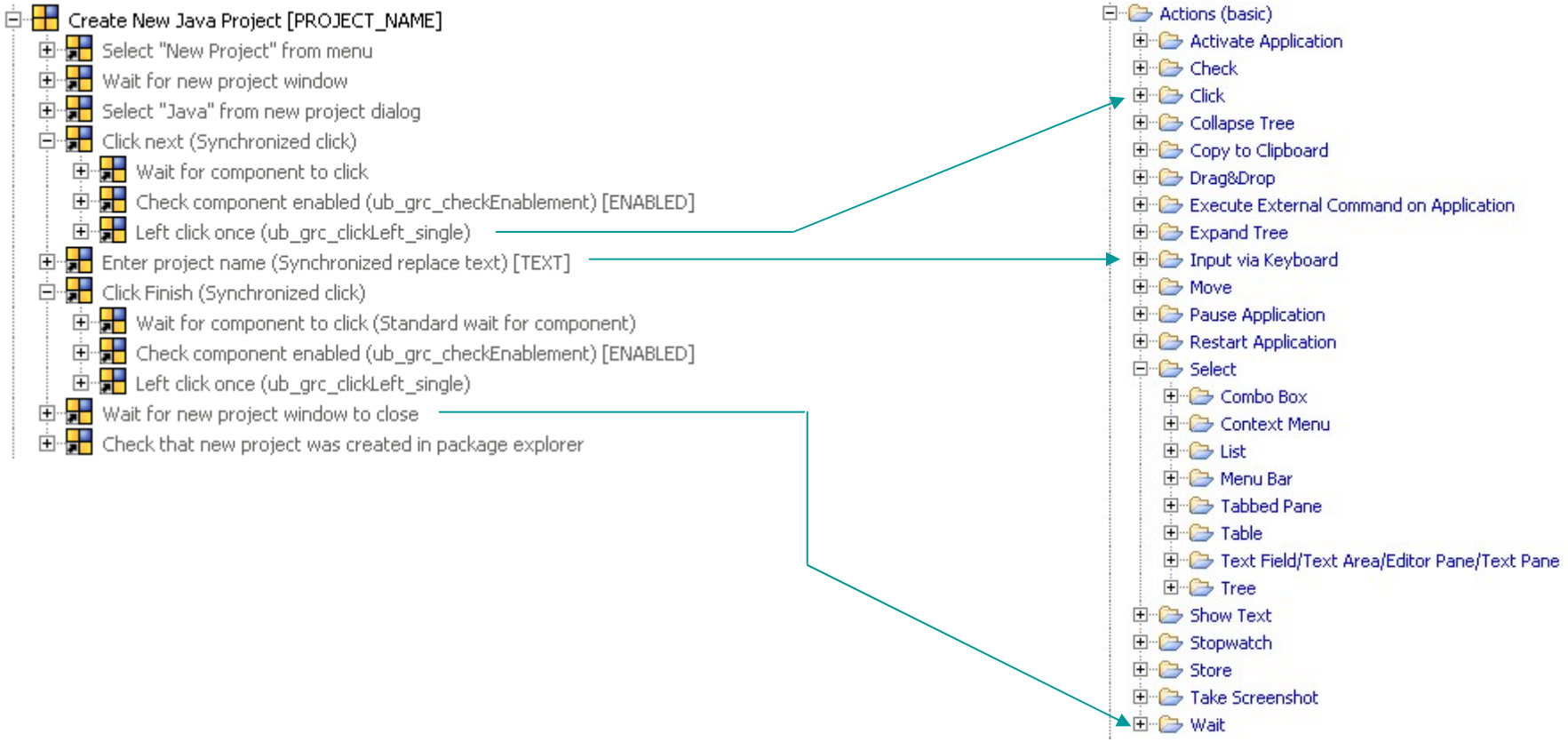
Effort to make maintainable

Tests *as-is* not *as-should-be*

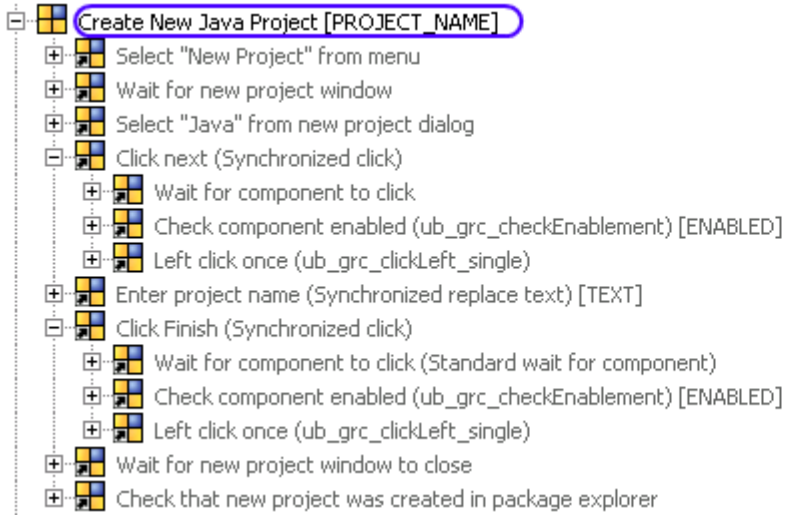
No explicit intelligence in test

Method 3 – Automated with keywords

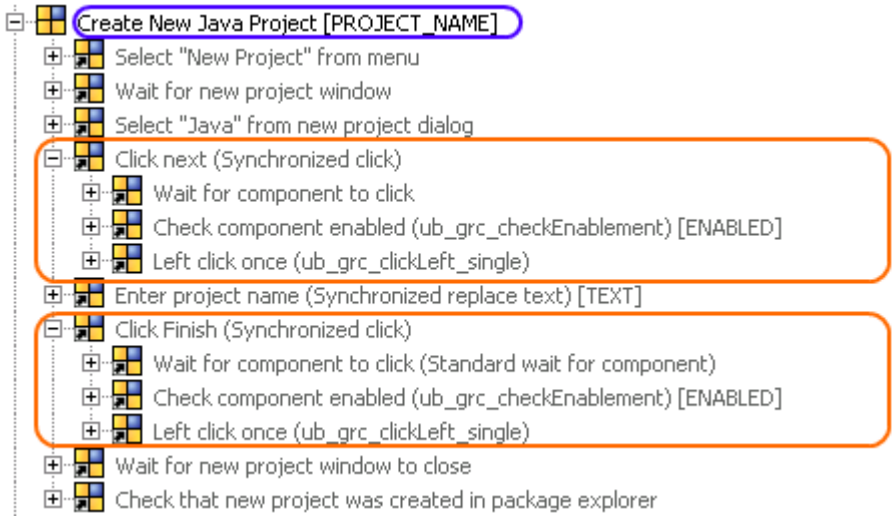
Method 3 – Automated with keywords



Method 3 – Automated with keywords



Method 3 – Automated with keywords



Method 3 – Automated with keywords

Positive aspects

High-level actions

Independence from app

Earlier focus on acceptance

Readable tests

No programming effort

Modularity almost for free

Tester in full control

Negative aspects

Requires planning & design

Requires discipline:

→ naming conventions

→ any restructuring work

→ management of keywords

Method 3 – Automated with keywords

- Positive aspects
  - High-level actions
  - Independence from app
  - Earlier focus on acceptance
  - Readable tests
  - No programming effort
  - Modularity almost for free

- **"Negative" aspects**
  - Requires planning & design
  - Requires discipline:
  - → naming conventions
  - → any restructuring work
  - → management of keywords

- **No different to development!**

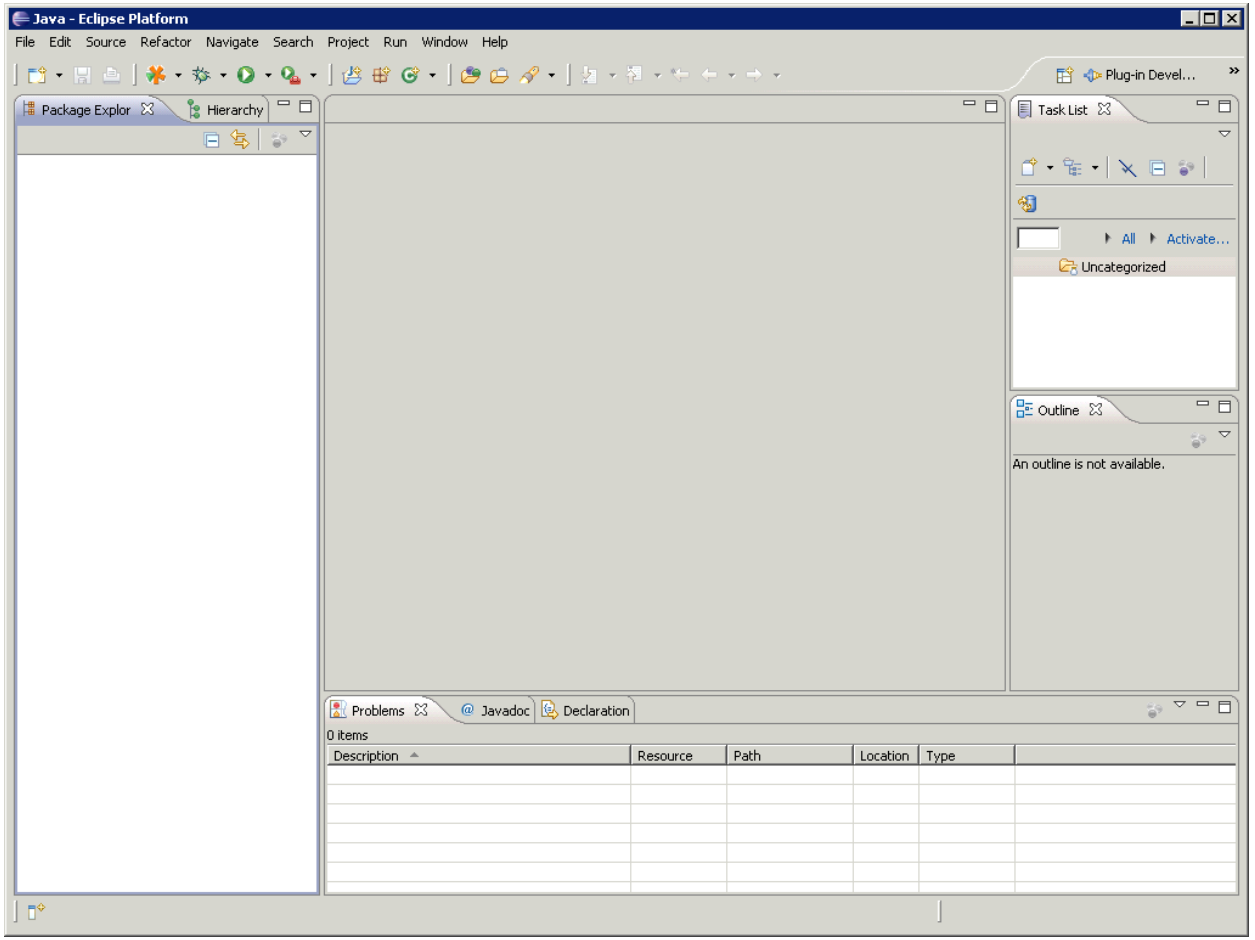
Automatic tests are as important as code!

- **Must be planned**
  - Which areas should be tested
  - Which areas should be automated
- **Must be structured**
  - Based on clear use case
  - Test design
  - Reusable, flexible modules
  - Readable and maintainable
- **Contain explicit intelligence**
  - Synchronization
  - Checks and fall backs
  - Knowledge of application
  - Understanding of controls
  - Understanding of workflows

Automatic tests are as important as code!

- **Must be planned**
  - Which areas should be tested
  - Which areas should be automated
- **Must be structured**
  - Test design
  - Reusable, flexible modules
  - Readable and maintainable
- **Contain explicit intelligence**
  - Synchronization
  - Checks and fall backs
  - Knowledge of application
  - Understanding of controls
  - Understanding of workflows
- **Keywords → well structured tests without coding effort**

Keyword-driven test



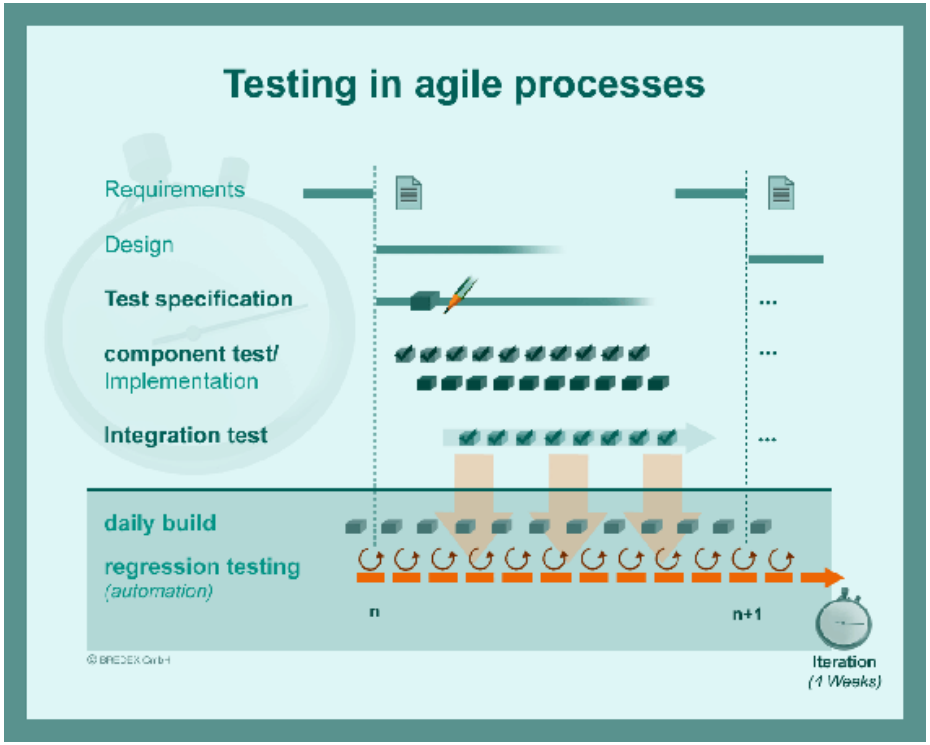
Getting the mix right – automated tests

- **Daily**

  Daily build – smoke test

- **Weekly**

  Complete automated test



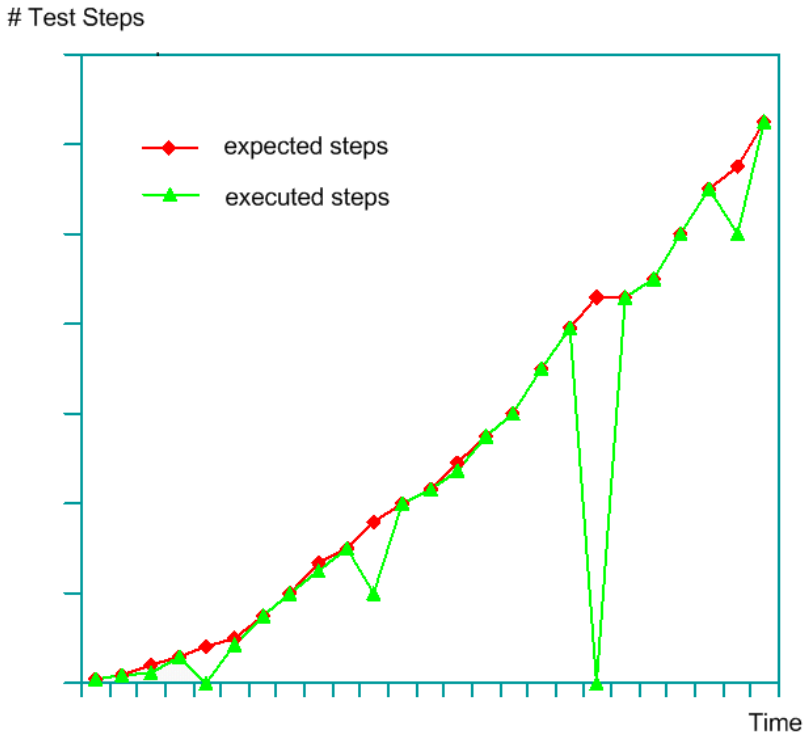
Getting the mix right – automated tests

- **Daily**

  Daily build – smoke test

- **Weekly**

  Complete automated test



Getting the mix right – manual tests

- **Daily**

"Four-eye" test

- **Weekly**

Show and tell

- **Per feature (timely)**

Manual acceptance test

- **Before release**

Full manual test phase

Results

- Acceptance tests at the forefront of development
- Timely discussion and time to react
- Customer satisfaction