Best practices for the creation of automated agile GUI tests

Markus Tiede

Bredex GmbH

 Version 2.4

About me

- Markus Tiede
- Student from Germany (8th semester)
- For 2 years working for Bredex GmbH in Braunschweig
- Designing and automating GUI tests
- Including tests for Java (swing, swt), Eclipse and Eclipse based applications (rcp) and html websites

Aims/Agenda

Features of agile processes

Aims/Agenda

- Features of agile processes
- Requirements for agile GUI tests

Aims/Agenda

- **Features of agile processes**
- **Requirements for agile GUI tests**
- **Resolutions**

  What do we have to do?

Aims/Agenda

- **Features of agile processes**
- **Requirements for agile GUI tests**
- **Resolutions**
  - What do we have to do?
- **Test design**
  - How do we do it?

Aims/Agenda

- **Features of agile processes**
- **Requirements for agile GUI tests**
- **Resolutions**
  - What do we have to do?
- **Test design**
  - How do we do it?
- **How design fulfills requirements**

Aims/Agenda

- **Features of agile processes**
- **Requirements for agile GUI tests**
- **Resolutions**
  - What do we have to do?
- **Test design**
  - How do we do it?
- **How design fulfills requirements**
- **A project example**

Features of agile processes

Features of agile processes

Requirements

Not defined in detail

Subject to change

Features of agile processes

- **Requirements**
  - Not defined in detail
  - Subject to change
- **Frequent and early releases**

Features of agile processes

- **Requirements**
  - Not defined in detail
  - Subject to change
- **Frequent and early releases**
- **Continuous testing**

Features of agile processes

- **Requirements**
  - Not defined in detail
  - Subject to change
- **Frequent and early releases**
- **Continuous testing**
- **Communication over documentation**

Requirements for agile GUI tests

- Need to match test and development processes

Requirements for agile GUI tests

- **Need to match test and development processes**
- **Flexibility**

  Because requirements (and the GUI) will change

Requirements for agile GUI tests

- **Need to match test and development processes**
- **Flexibility**
  - Because requirements (and the GUI) will change
- **Test creation in parallel with development**
  - Integration tests written early and quickly
  - … before the GUI is available

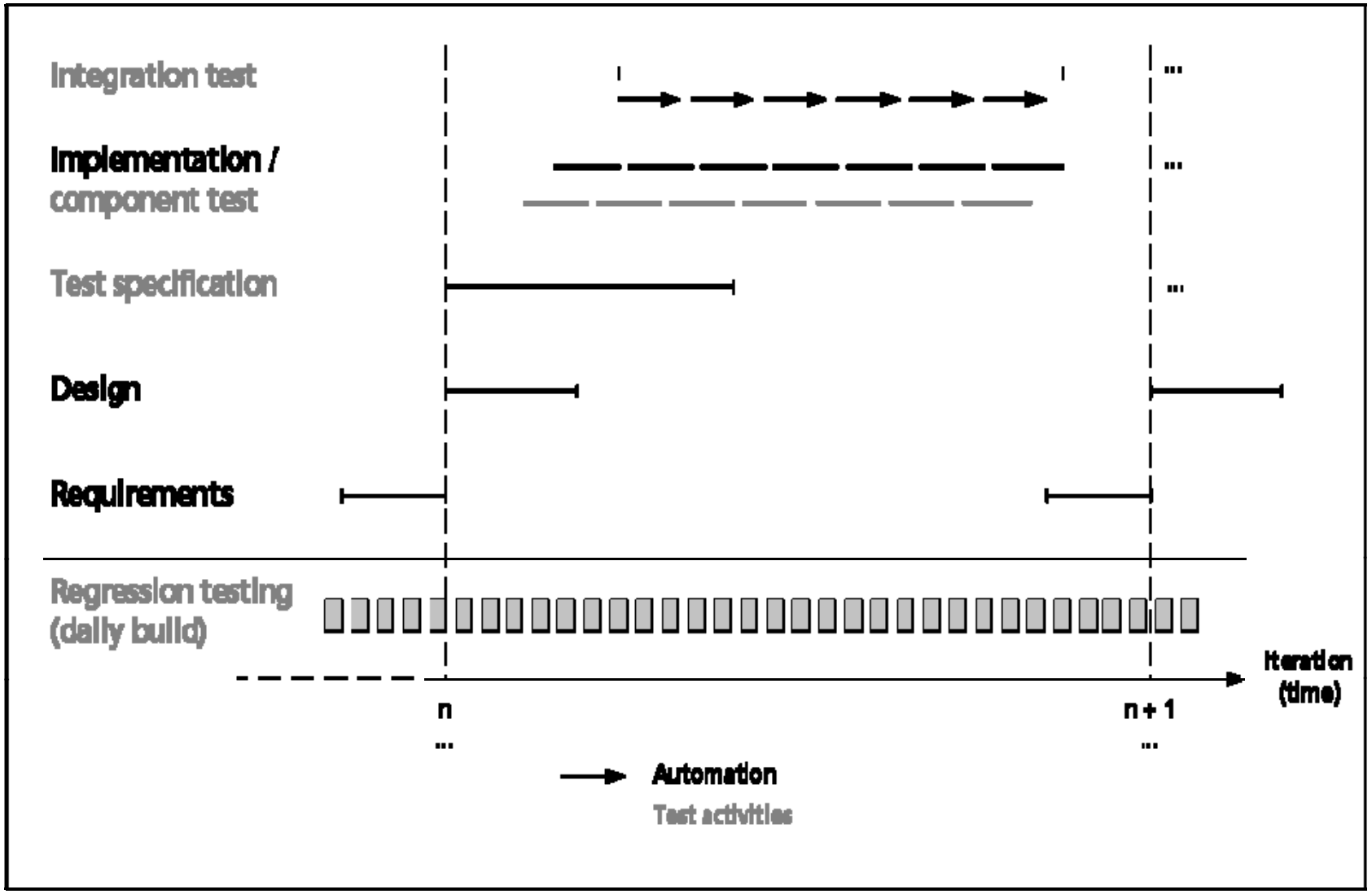
Requirements for agile GUI tests

- **Need to match test and development processes**
- **Flexibility**
  Because requirements (and the GUI) will change
- **Test creation in parallel with development**
  Integration tests written early and quickly
  … before the GUI is available
- **Stability of tests**
  Low maintenance effort for regression tests

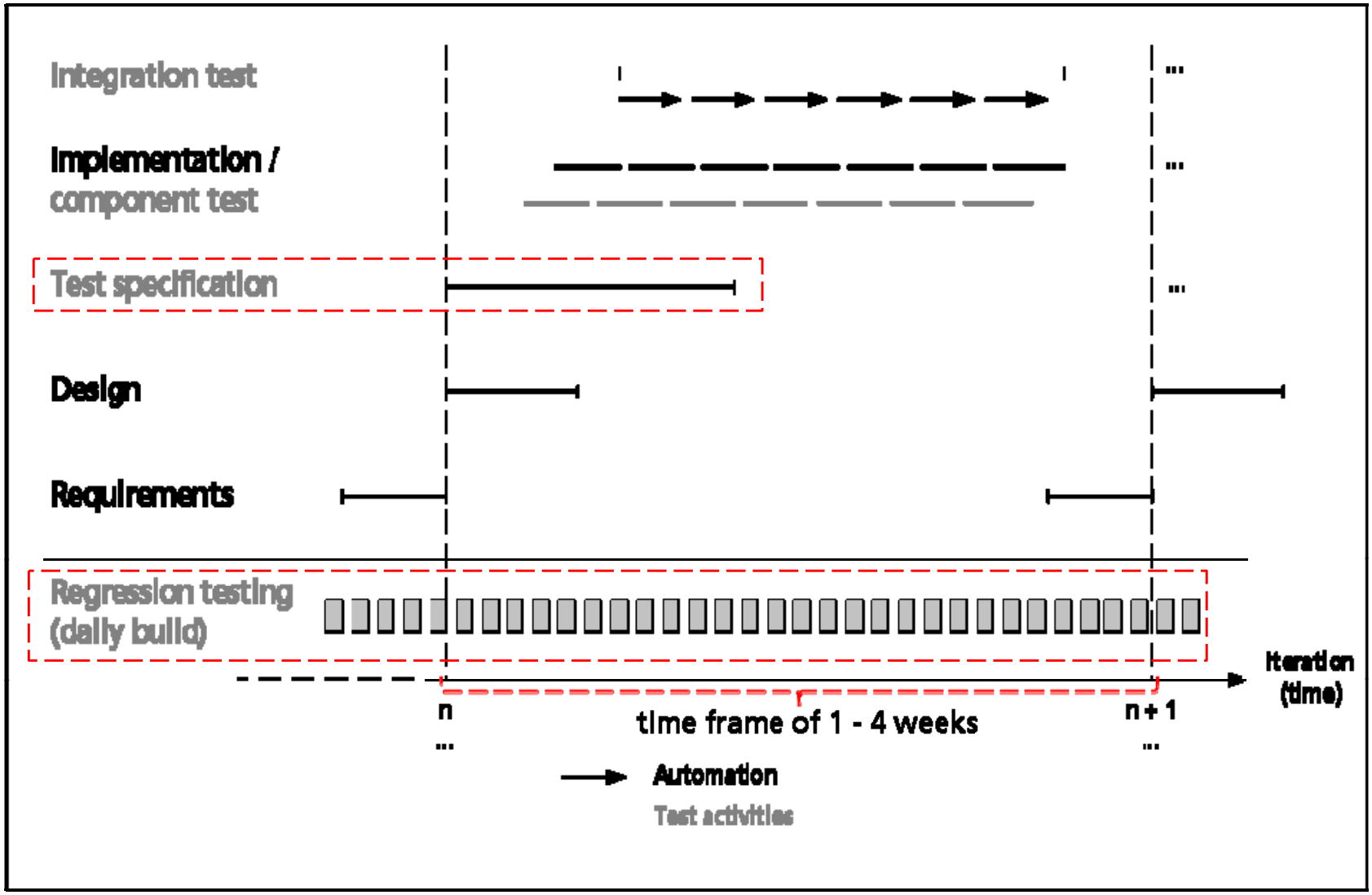
Requirements for agile GUI tests

- **Need to match test and development processes**
- **Flexibility**
  Because requirements (and the GUI) will change
- **Test creation in parallel with development**
  Integration tests written early and quickly
  … before the GUI is available
- **Stability of tests**
  Low maintenance effort for regression tests
- **… Test early, test often, test efficiently**

Testing agilely



Testing agilely



Resolutions – test early

Independence from application

Write tests from beginning

Based on requirements *for this sprint*

Basic knowledge of expected interface (inputs / outputs)

Resolutions – test early

- **Independence from application**
  - Write tests from beginning
  - Based on requirements *for this sprint*
  - Basic knowledge of expected interface (inputs / outputs)
- **Well-planned test design**
  - Levels of abstraction
  - What *kinds* of things have to be tested?

Resolutions – test often

Integration tests

Easy and quick to create

Automate easily and quickly

Resolutions – test often

- **Integration tests**
  - Easy and quick to create
  - Automate easily and quickly
- **Regression tests**
  - Stable
  - Easy to maintain

Resolutions – test often

- **Integration tests**
  - Easy and quick to create
  - Automate easily and quickly
- **Regression tests**
  - Stable
  - Easy to maintain
- **Teamwork**
  - Whole test team involved in test process

Resolutions – test efficiently

Long life span for tests

Resolutions – test efficiently

- **Long life span for tests**
- **Flexibility**
  - Data
  - Timing
  - Implementations / platforms
  - GUI components / layout
  - Test flow

Test design

Abstract tests

Data, GUI component placeholder, test specification separate from implementations

Parametrize data

Use more general specifications where possible

Test design

Abstract tests

Data, GUI component maps, implementations separate from test specification

Parametrize data

Use more general specifications where possible

Modular tests

Create basic modules for reusability

“Do it once and only once!”

Examples

Abstract tests

simple application: simple adder

Test modules

Changing behavior
Synchronization

Late-coming features
Login dialog

Example 1.1: simple adder

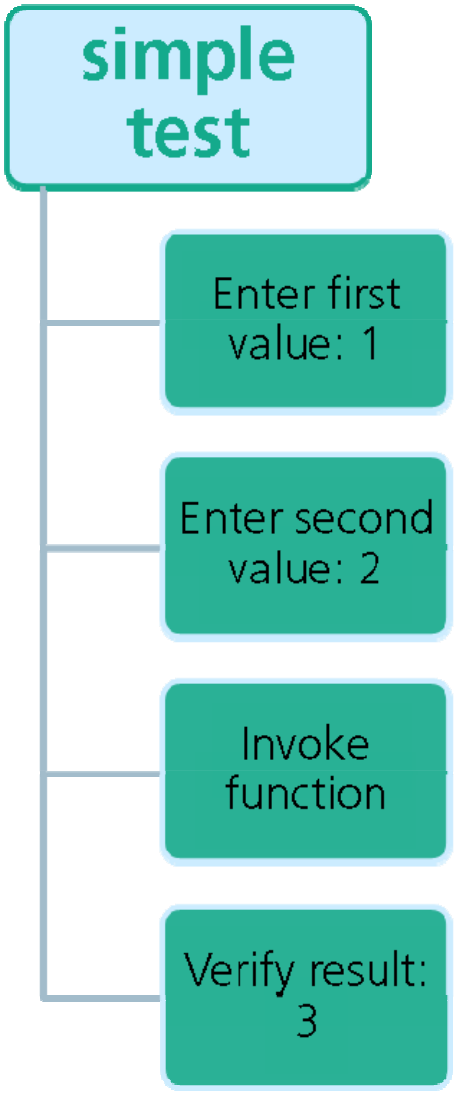
Functional Specification of program

- Sum up two user defined numbers
- User can invoke sum function
- Show result on screen

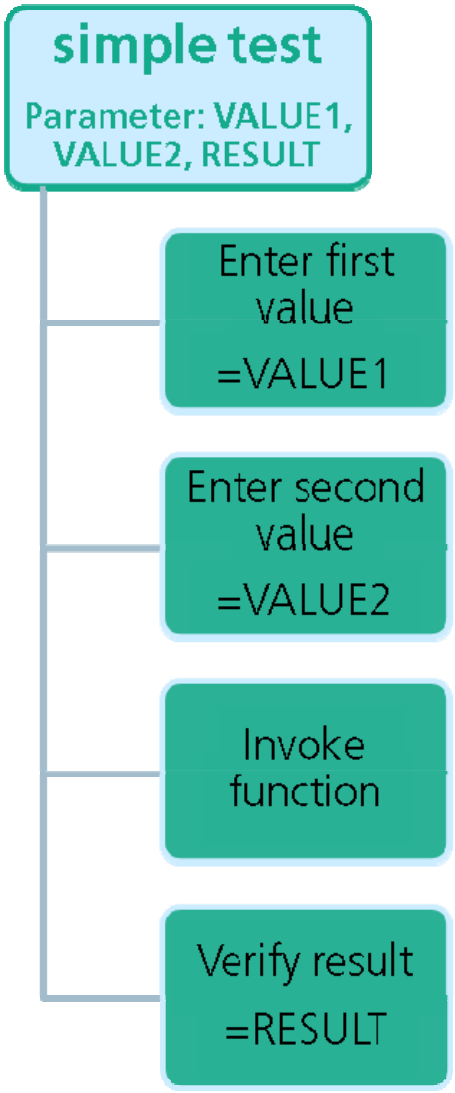
Simple test

- Enter first value: “1”
- Enter second value: “2”
- Invoke function
- Verify result: “3”

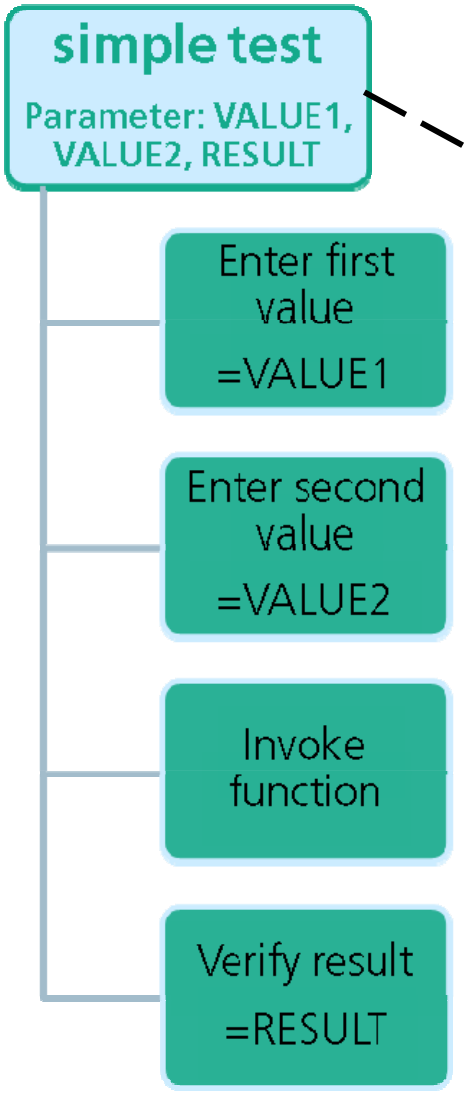
Example 1.2: simple adder



Example 1.3: simple adder – separate Data



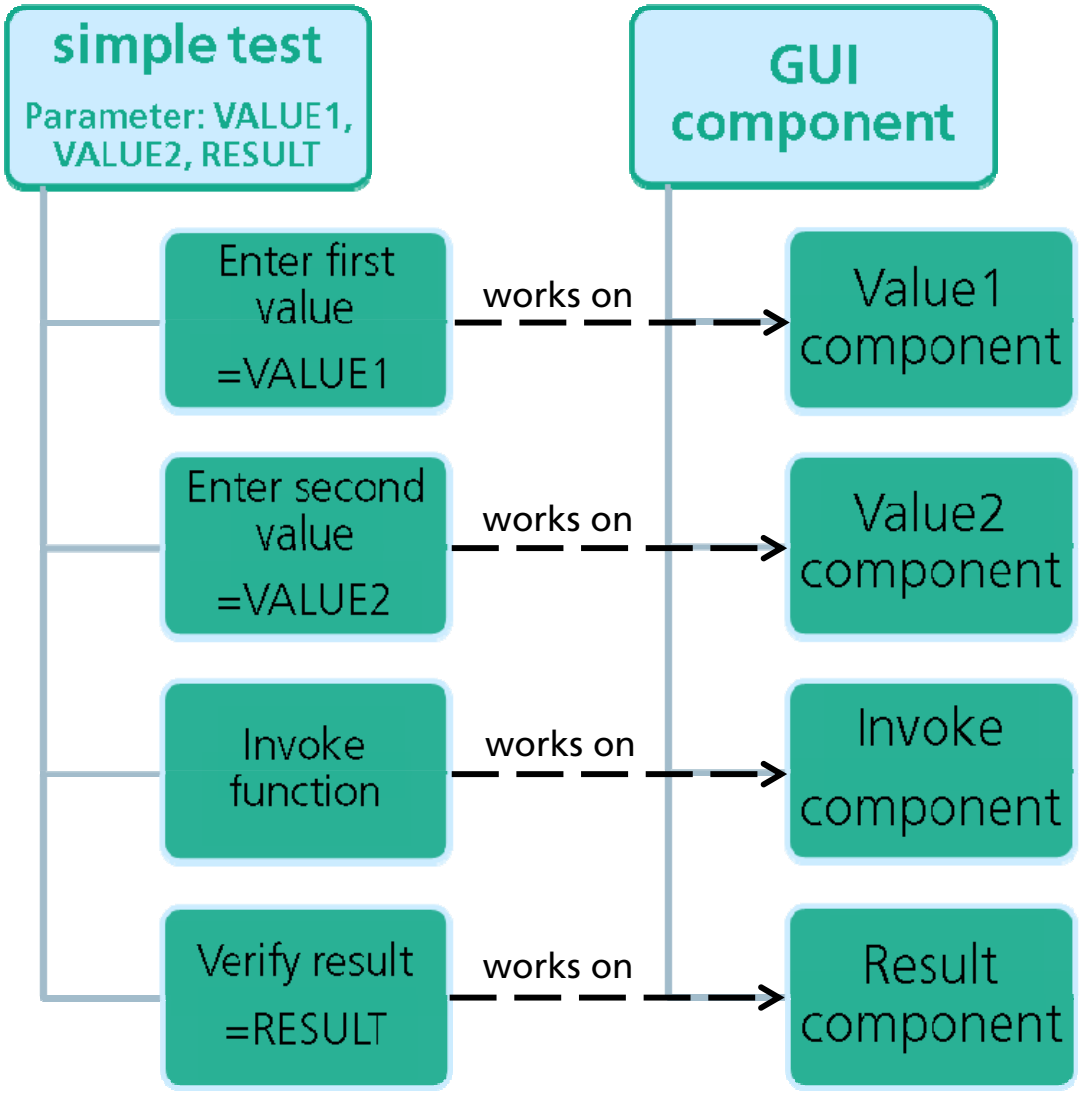
Example 1.3: simple adder – separate Data



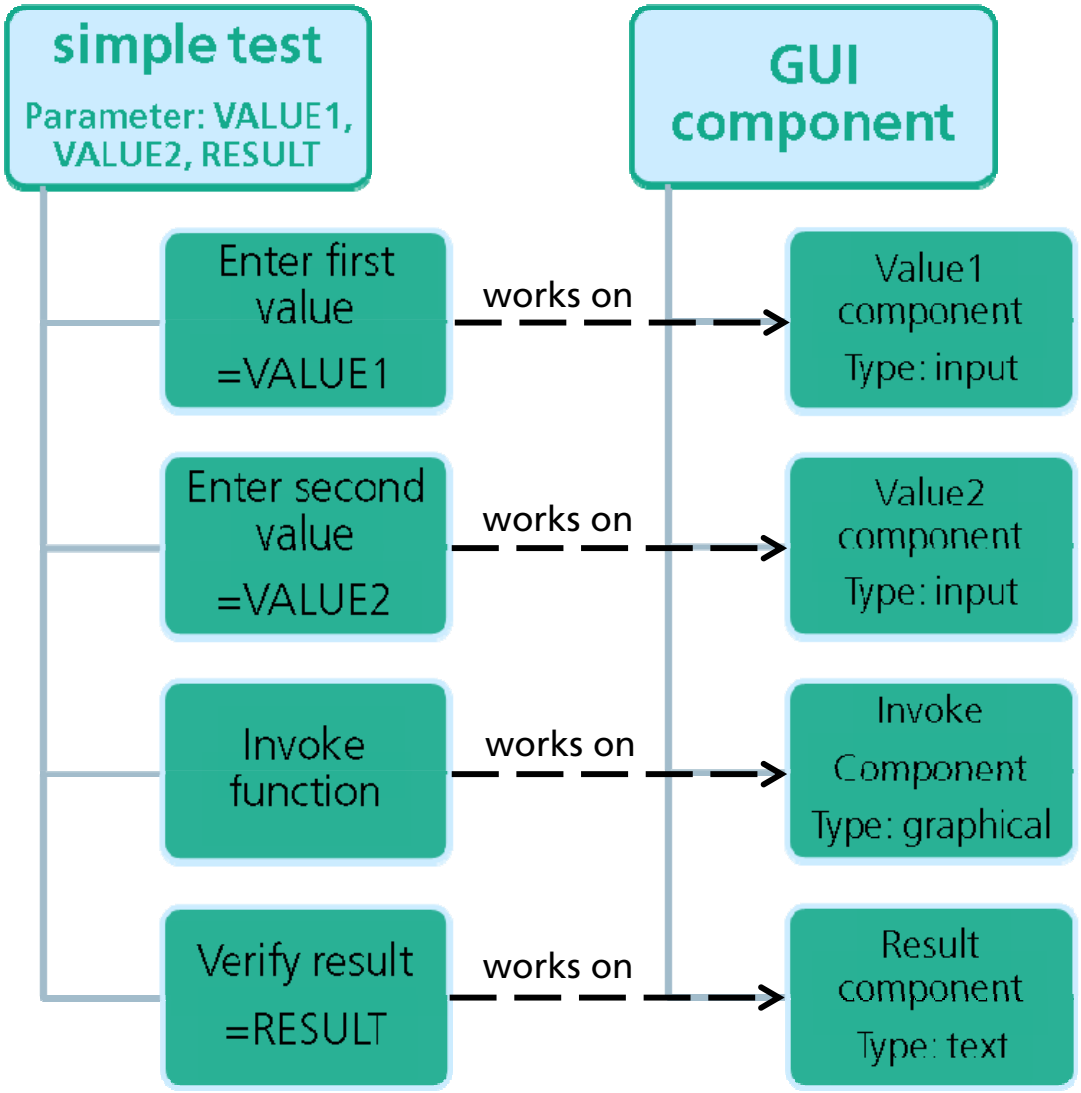
refers to

VALUE1	VALUE2	RESULT
1	2	3
1	-1	0
17	4	21

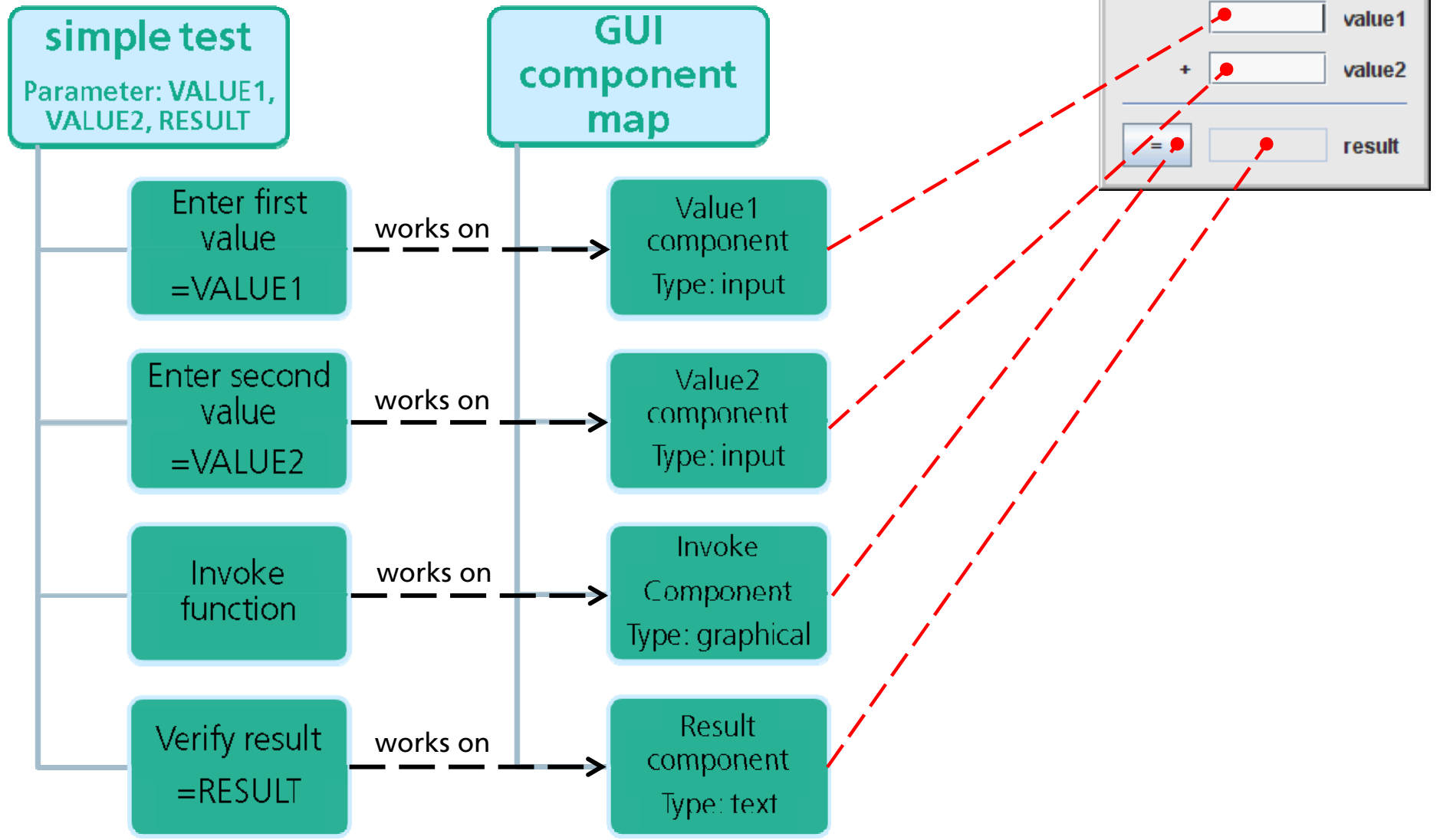
Example 1.4: simple adder – separate GUI components



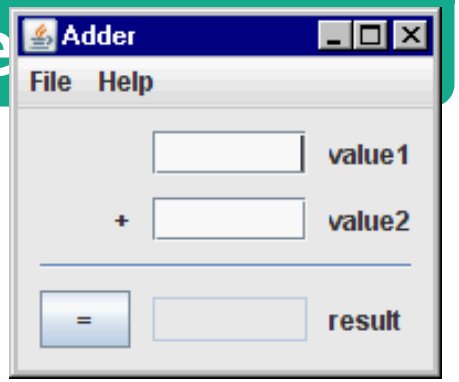
Example 1.5: simple adder – use generic types

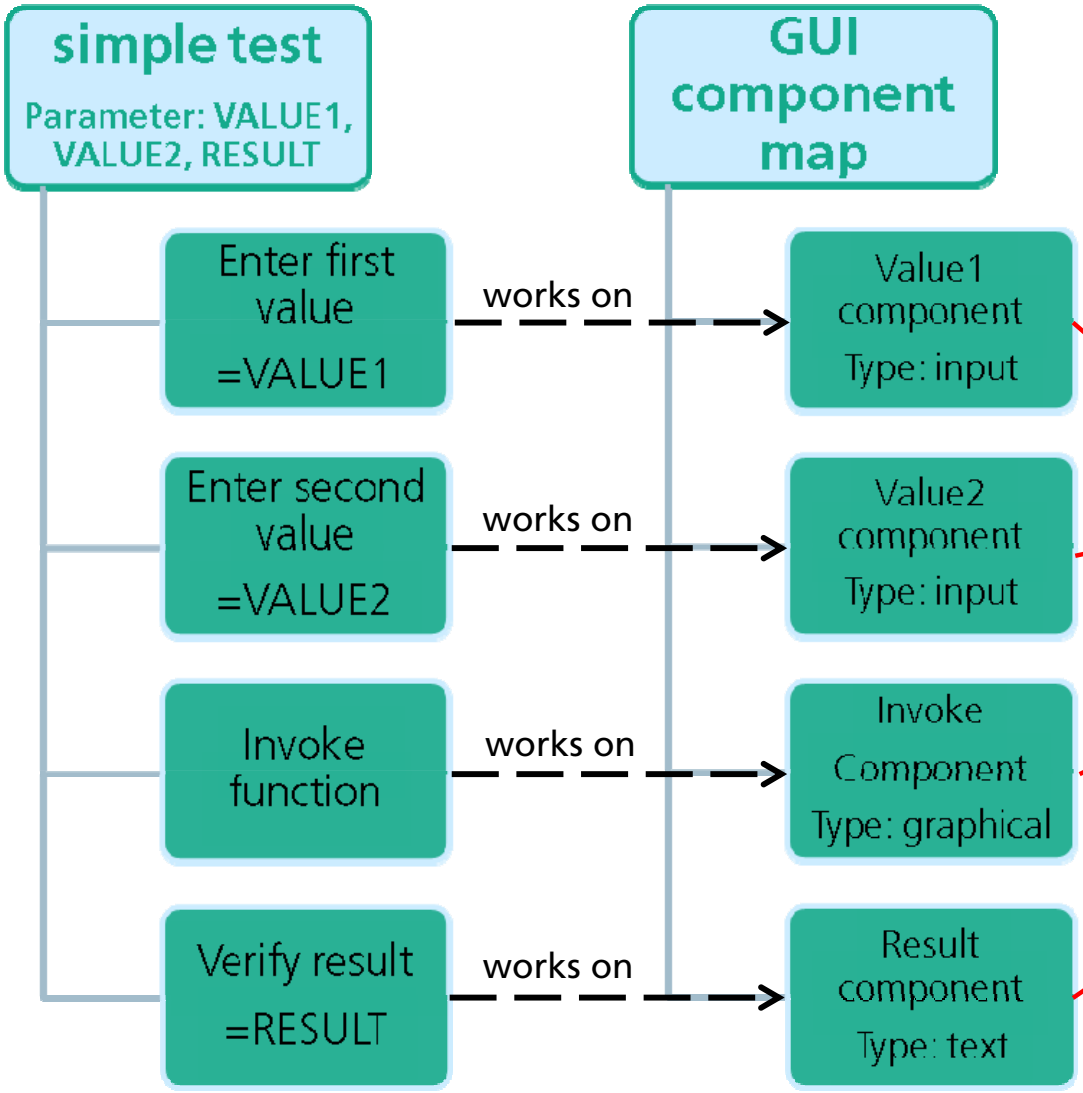


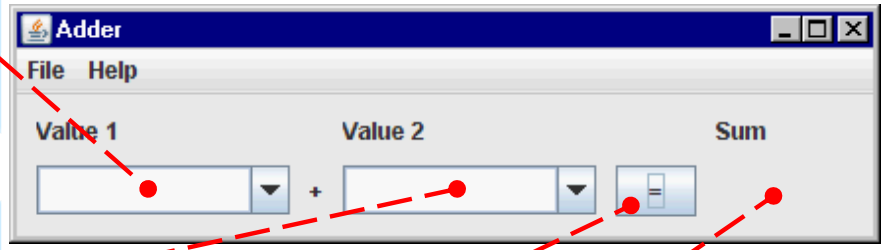
Example 1.6: simple adder – use ge



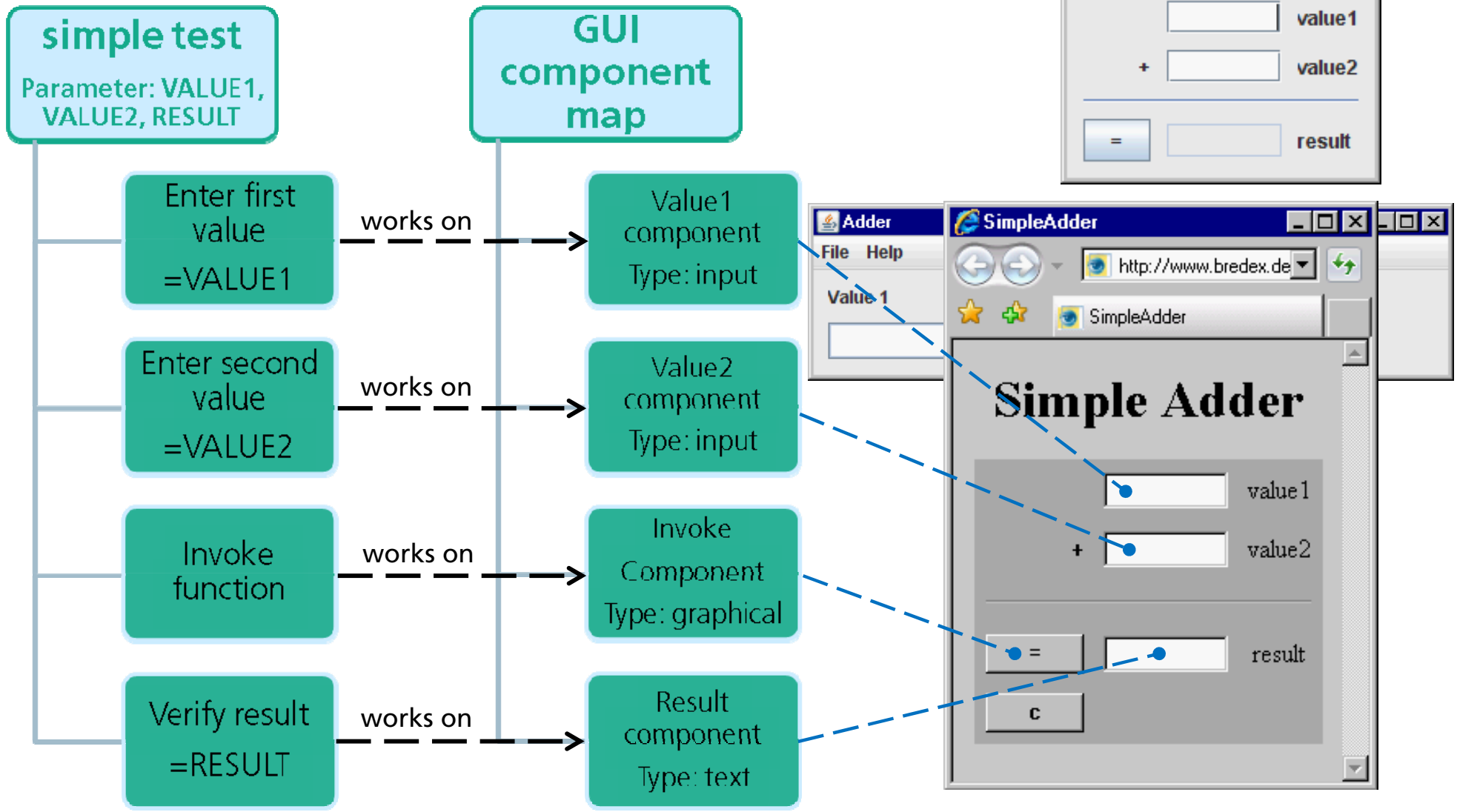
Example 1.6: simple adder – use ge







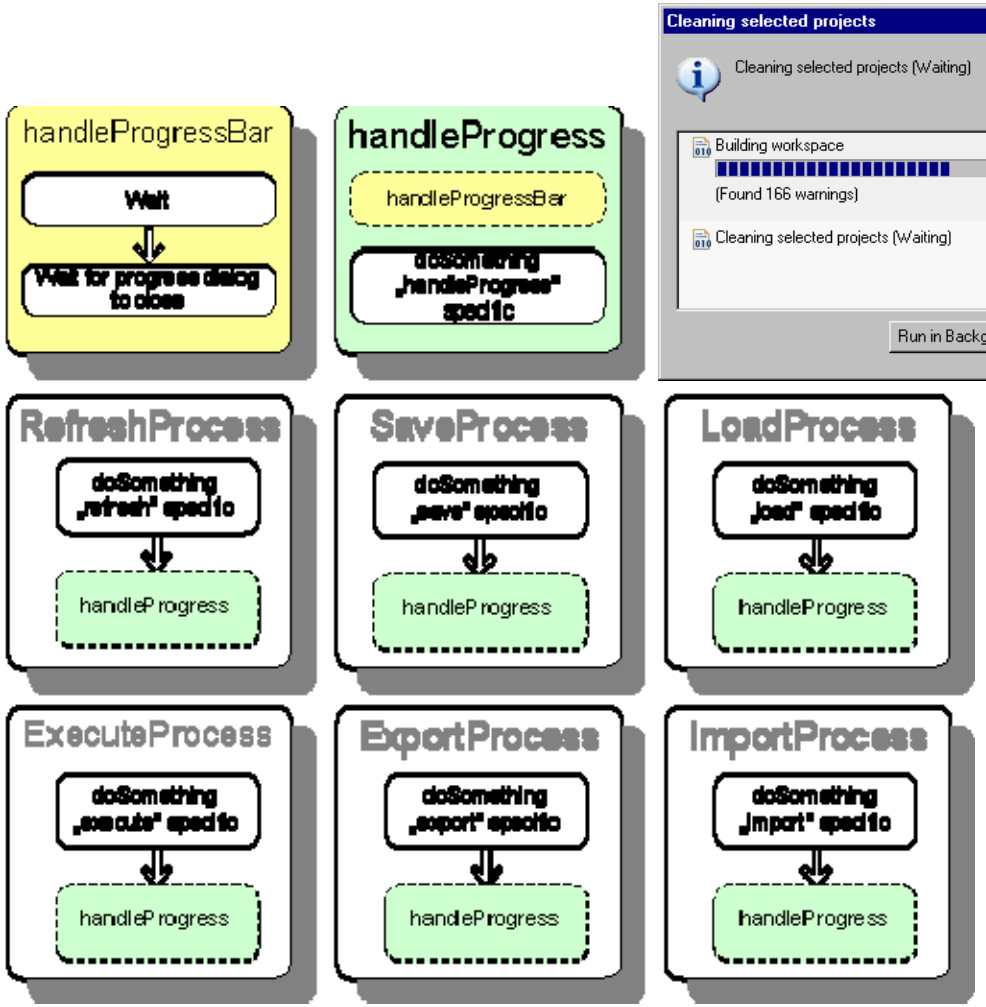
Example 1.6: simple adder – use ge



Example 2.1: Synchronization

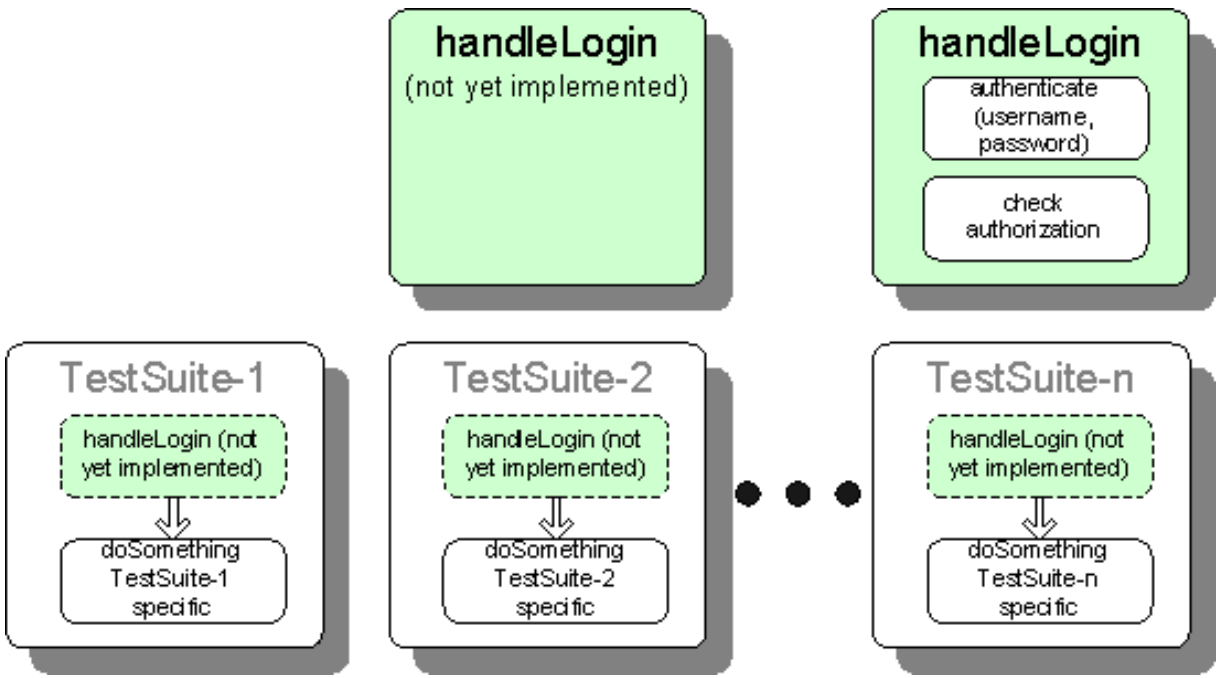
- previous behavior
- *always* appears for longer lasting actions
- simple synchronization during test flow:
  1. wait for dialog to appear
  2. wait for dialog to close

Example 2.2: Synchronization



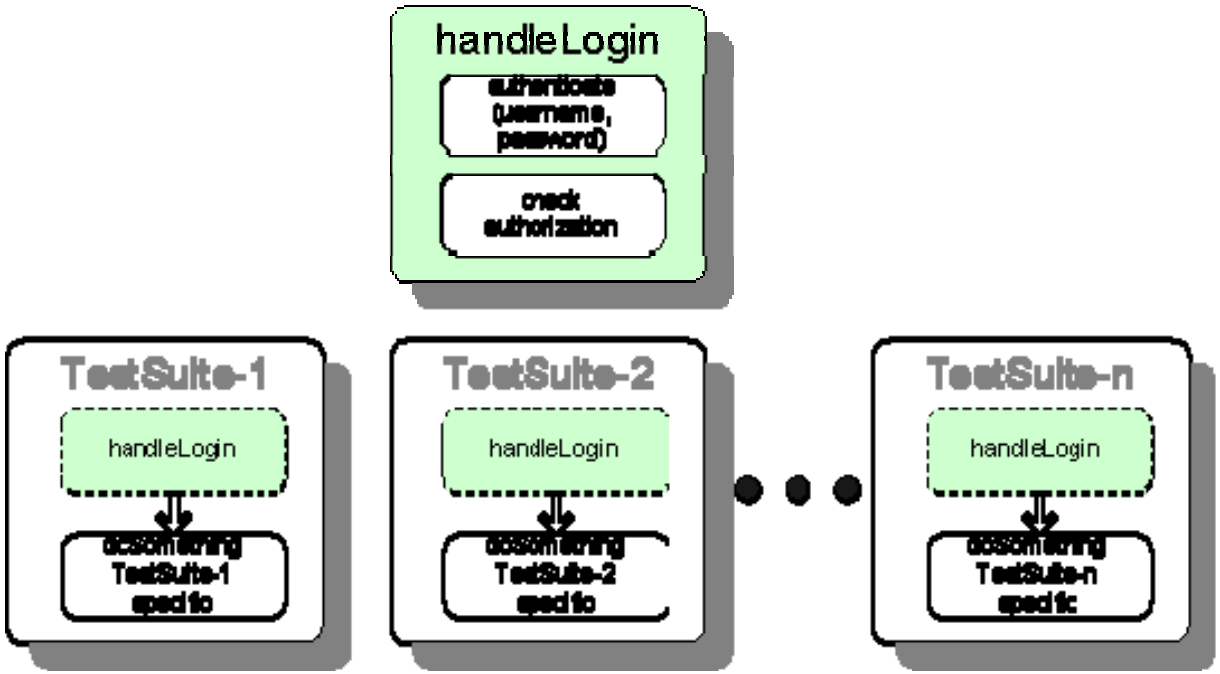
- new behavior
- only appears if action lasts longer than 1 second
- -> adjust module for the handling of progressBar

Example 3.1: Login dialog



- Feature scheduled for later release e.g. login mechanism
- consider in test design
- e.g. use empty modules as placeholder

Example 3.2: Login dialog



- as soon as mechanism is available
- fill placeholders with life

How test design fulfills requirements

Abstract tests

Independence from application

Based on requirements and communication

Flexibility in details

Modular tests

Library of test modules

Centralized updates

So...

Using abstract modules in your test design supports agile GUI testing because tests can be written based on minimal details and are easily adaptable to changing plans.

Project figures

Twice-weekly releases

Often with GUI changes

One complete GUI framework update

Project figures

- **Twice-weekly releases**
  - Often with GUI changes
  - One complete GUI framework update
- **Tests**
  - Library of modules

Test duration:	9 hours
Maintenance time:	1 hour/week
Cost:	<10% of overall project cost

Summary

Agile development requires agile testing

Summary

- **Agile development requires agile testing**
- **Agile GUI testing is possible**
  - Must be automated

Summary

- **Agile development requires agile testing**
- **Agile GUI testing is possible**
  - Must be automated
- **Test design is important**

Summary

- **Agile development requires agile testing**
- **Agile GUI testing is possible**
  - Must be automated
- **Test design is important**
- **Agile testing:**
  - Improves quality – continual monitoring
  - Can reduce cost of automated testing

Thank you for listening…

And for any outstanding questions…

… now is the time to ask them

… or visit me at stand 10 in the demonstration hall