Automated Software Validation

*Study manual*

Workload : 4 EC

Date of release : 08 Nov 2021

Academic year : 2021 - 2022

Course : Automated Software Validation

Module coordinator : Karthik Srinivasan

Semester : Software Engineering

**Table of Contents**

[1 Introduction 3](#_Toc101186799)

[2 Learning outcomes 4](#_Toc101186800)

[3 Programme 5](#_Toc101186801)

[3.1 Content 5](#_Toc101186802)

[3.2 Methods 5](#_Toc101186803)

[3.3 Study materials 5](#_Toc101186804)

[3.3.1 Lecturer Slides & Materials (BrightSpace/DLO) 5](#_Toc101186805)

[3.3.2 Book 5](#_Toc101186806)

[3.3.3 Video materials 5](#_Toc101186807)

[3.3.4 Online material 5](#_Toc101186808)

[3.4 Software Project Selection Guidelines 6](#_Toc101186809)

[3.5 Lecturers and accessibility 7](#_Toc101186810)

[3.6 Planning 7](#_Toc101186811)

[4 Examination and assessment 10](#_Toc101186812)

[4.1 Introduction 10](#_Toc101186813)

[4.2 Assignment 10](#_Toc101186814)

[4.2.1 Assignment, part 1 10](#_Toc101186815)

[4.2.2 Assignment, part 2 10](#_Toc101186816)

[4.2.3 Assignment, part 3 11](#_Toc101186817)

[4.3 Grading rubric 11](#_Toc101186818)

[4.4 Exam results 11](#_Toc101186819)

# Introduction

The time to market, always plays a key role in everyday business, the shorter the time the better is the business climate. This is made possible by the use of Continuous Delivery technique applied alongside the SCRUM/Agile practice.

Continuous Delivery using the Continuous Integration pipeline is effective in delivering business value among the Software development and maintenance fields of business. Most important topics for the establishment of CI-Pipeline and maintenance of the software are covered in this course.

# Learning outcomes

The objective of this course is to apply and realize different techniques and methods available for;

1. Software Requirements engineering
2. Software Testing
3. Software Maintenance

on a chosen software project based on guidelines.

While maintaining the focus on the above-mentioned topics, adherence towards the Scrum/Agile methodology will be practiced while iterating the sprints for Continuous Delivery via a CI-Pipeline setup.

Main course objective breaks down into the following learning outcomes.

* To familiarize with requirements analysis, gathering, and documentation.
* To familiarize various test principles and their applicability.
* To understand & write unit tests to verify & validate the requirements.
* To familiarize analyzing the code & suggesting recommendations.
* To apply & evaluate the application, with different software maintenance techniques.
* To execute & familiarize the concept of Continuous Delivery.
* To practice & work with Continuous Integration Pipeline.
* To realize the concept of code refactoring.

# Programme

## Content

Students will advance their skills in software requirements analysis, maintenance and testing. Students will also improve their knowledge on continuous integration and continuous delivery process.

## Methods

In this course, hands on workshops are mixed with feedback sessions for assignments. In the workshops, we will discuss theory and work on small exercises to clarify key concepts. The feedback sessions are scheduled to discuss the submitted assignments.

## Study materials

### Lecturer Slides & Materials (BrightSpace/DLO)

* Lecture slides Theory and Practice
* SIG PDF

### Book

* Pragmatic Unit Testing in Java 8 with Junit
* Building Maintainable Software, Java Edition, 2016, O’Really Media  
  J. Visser, S. Rigal, R. van der Leek, P. van Eck, G. Wijnholds
* A Practical Model for Measuring Maintainability  
  I. Heitlager, T. Kuipers, J. Visser

**Use case**

* Managing Software Requirements: A Use Case Approach, Second Edition, Addison-Wesley Professional, May 2003, ISBN: 9780321122476
* Use Case Modelling, Addison-Wesley Professional, August 2002, ISBN: 9780201709131

O’Really Media, lib.hva.nl

* User Story Mapping, O'Reilly Media, Inc. September 2014, ISBN: 9781491904909.

**Requirements**

* Mastering the Requirements process: Getting Requirements Right, 3/e, Video Enhanced Edition, Suzanne Robertson & James Robertson, Addison-Wesley Professional, 2012.

### Video materials

* Writing User Stories, USER EXPERIENCE (UX), Ryan Harper, O'Reilly Media, Inc., September 2017

### Online material

* [Source meter]
  + <https://www.sourcemeter.com/>
* [Design Complexity]
  + <https://www.tutorialspoint.com/software_engineering/software_design_complexity.htm>
* [Lynda]
  + <https://www.lynda.com/>
* [Vogella] Junit
  + <http://www.vogella.com/tutorials/JUnit/article.html>
* [Vogella] Mockito:
  + <http://www.vogella.com/tutorials/Mockito/article.html>
* [SonarQube] SonarQube:
  + <https://www.sonarqube.org/>
* [Jenkins] Jenkins
  + <https://jenkins.io/>

## Software Project Selection Guidelines

There are a number of requirements that must be met by a software project in order for it to be acceptable for the practical lab of the automated software validation course. Besides some easy to check requirements there are also some more challenging requirements that must be met. In order for students to determine, if their project meets the later one’s, students need to analyze their project using SonarQube (http://www.sonarqube.org). SonarQube is an open source static code analyzer. Once students have installed SonarQube they must configure their build tool to instruct SonarQube to perform the actual analysis. Instruction for configuring Maven and running the analysis can be found [here](http://docs.sonarqube.org/display/SCAN/Analyzing+with+SonarQube+Scanner+for+Maven).

Note: Students can bring in their own project from Project SE, discuss it with your lecturer.

**Criteria for software project**

Each software project should meet the following criteria in order for it to be acceptable

* must be a project using a strongly typed language. (The main restriction is that SonarQube must be able to analyze the project.)
* must be build using some sort of build tool like Maven or Gradle.
* must be able to run stand-alone (e.g. without any interaction with external systems that prevent it from preforming basic operations). Libraries and frameworks are excluded.
* must have at least about 10.000 lines of code. In case of own projects, students have the possibility to discuss with their Lecturers.
* after analysis by SonarQube the following numbers must true
  + Number of bugs reported by SonarQube > 10
  + Number of vulnerabilities > 5
  + Number of Code Smells > 25, of which at least 2 indicating that a method or a class has a cyclomatic complexity >= 6

These numbers represent the minimum values. It is up to the lecturer to set slightly higher values if he/she finds them a better fit for the project.

* Sources must be accessible for the lecturer (during the course).

Note:

Students can also approach their corresponding Lecturers to get an approval on a case-by-case basis.

**Required software**

During the practical labs students will be using the following software’s, so make sure these are installed on your system.

* Java
* Maven (http://maven.apache.org/) - Build tool
* SonarQube (http://www.sonarqube.org/) - Static code analyzer
* Jenkins (https://jenkins.io/) - Continuous integration server

**Sonar configuration**

Adjust the properties sonar.jdbc.username , sonar.jdbc.password and

sonar.jdbc.url to match those of your database in

${SONAR\_HOME}/conf/sonar.properties.

**Maven configuration**

Configure maven either on a project level or system level. The later can be done

by adding the configuration in ${HOME}/.m2/settings.xml.

**Possible candidates**

https://github.com/UweTrottmann/SeriesGuide

https://sourceforge.net/projects/firemox/

https://github.com/umlet/umlet

https://github.com/C-Aniruddh/ACEMusicPlayer

https://github.com/DrKLO/Telegram

<https://github.com/siacs/Conversations>

Students are advised to choose a project that is recent. Check the commit status.

## Lecturers and accessibility

Module coordinator: Karthik Srinivasan, [k.srinivasan@hva.nl](mailto:k.srinivasan@hva.nl)

Lecturer: Nick van Leeuwen, [n.a.van.leeuwen@hva.nl](mailto:n.a.van.leeuwen@hva.nl)

Lecturer: Evert Jan Couperus, [e.j.t.couperus@hva.nl](mailto:e.j.t.couperus@hva.nl)

Lecturer: Karthik Srinivasan, [k.srinivasan@hva.nl](mailto:k.srinivasan@hva.nl)

## Planning

|  |  |  |  |
| --- | --- | --- | --- |
| **W** | **T/P** | **CONTENTS** | **READING (Recommended)** |
| 1 | T | Use Case, Functional & Non-Functional Requirements  Project Selection  Software & Tools | Managing Software Requirements: A Use Case Approach, Second Edition, Addison-Wesley Professional, May 2003, ISBN: 9780321122476 |
|  | P | **Student Activity:**  Selects his/her team project  Perform an analysis using SonarQube on the project  Request approval from Lecturer for the selected project   * Check the Software Selection Guideline * Use results from SonarQube to discuss with corresponding lecturer(s).   Install, Configure, Set up Project environment, Software, and necessary tools.  Draft Use Case, functional and non-functional requirements. | Mastering the Requirements process: Getting Requirements Right, 3/e, Video Enhanced Edition, Suzanne Robertson & James Robertson, Addison-Wesley Professional, 2012. |
| 2 | T | User Stories  CI/CD  SonarQube | User Story Mapping, O'Reilly Media, Inc. September 2014, ISBN: 9781491904909. |
| 2 | P | **Student Activity:**  Drafts user stories  Work on the Integration and delivery pipeline  **Deadline: Upload the PA1 document on DLO.** |  |
| 3 | T | Unit Test | Pragmatic Unit Testing in Java 8 with Junit |
|  | P | **Student Activity:**  Write Unit test |  |
| 4 | T | CI/CD (Jenkins, GITLab-HvA) | Refer to the Jenkins online documentation.  Refer to the Lecturer slides for HvA GITLab link, on the DLO. |
|  | P | **Student Activity:**  Complete the CI/CD pipeline  **Deadline: Upload the PA2 document on DLO.** |  |
| 5 | T | Software Maintenance | Building Maintainable Software, Java Edition, 2016, O’Really Media  J. Visser, S. Rigal, R. van der Leek, P. van Eck, G. Wijnholds |
|  | P | **Lecturer Activity**   * Feedback * Support for CI/CD |  |
| 6 | T | Software Maintenance  Software Quality Attributes, Metrics  SonarQube (Analysis)  SonarQube & ISO 25010 | A Practical Model for Measuring Maintainability  I. Heitlager, T. Kuipers, J. Visser |
|  | P | **Student Activity:**  Code Analysis  Refactor code |  |
| 7 | T | Catch-up |  |
|  | P | **Student Activity:**  Complete the assignment/report  **Deadline: Upload the PA3 document on DLO**. |  |

# Examination and assessment

## Introduction

Students are expected to work on a **two-member team** assignment that can be completed over the length of the course. The assignment is split up into smaller parts that focus on a certain topic and skills a student should master. Any deviations on the team size should be discussed with the class lecturer.

## Assignment

Students are instructed to upload **a file** containing the report as per the assignment instructions provided. Do include all project files and source code files required to compile and run your implementations on the **HvA GitLab** repository. **Link to source code must be included in the report.**

Deliver your .pdf file as per the format: <Project\_Name>.pdf

**Note:**

* Report the status of the Project before the start of the refactoring or testing
* SonarQube to be installed in advance.
* Student/Assignment Teams will be using the “**Activity: lines of code**” option on SonarQube to present their progress on the project.
* Student assignment reports on a given or chosen project can only be used (submitted for grading) for **one academic year**, which includes a **main** chance and a **resit** chance.
* Student assignment reports without GITLab link for their repository, will not be graded. They can use resit chance for re-submission.
* Issue Maintainer access on GITLab for your Lecturer(s).

### Assignment, part 1

**20% of the final result is based on the requirements part (2 points out of 10)**

* Document Product Use Cases
* Document User Stories
  + Card
  + Conversation
  + Confirmation

### Assignment, part 2

**30% of the final result is based on the test part (creation / adaptation of tests) (3 points out of 10)**

* Document the Application of test principles (First etc.)
* Document the build server setup and build history analysis
* Provide a diagram to show the CI/CD setup.
* Provide the Code snippets of implemented unit tests
* Document the Instructions to get the source code and set-up of the project.
* Document the traceability matrix mapping the user-stories and the tests.

### Assignment, part 3

**40% of the final result is based on the maintenance aspect (measure/refactor/measure and explain) (4 points out of 10)**

* Document the application SonarQube Criteria(s)
* Document the Measurements and metrics before refactoring
* Document the Analysis and recommendations for refactoring
* Document the Refactoring (code snippets)
* Document the Measurements and metrics after refactoring
* Document the Recommendations based on lessons learned, and for further improvement

**Note: 10% for language, style, layout etc. (1 point out of 10)**

**Also, refer to grading matrix for more information about the assignment requirements.**

## Grading rubric

Refer to the grading matrix/ attached rubric on DLO.

## Exam results

Grades are published via SIS. After the publication of the results, students will be offered a chance to look at their work and the comments made about their work.

If a student has grounds to appeal against the assessment, they may contact the lecturer to voice their objections within 25 working days after the publication of the grade. After this period, the possibility to change their grade will expire and they will have to re-participate in the examination.