# PA1489 Assignment Descriptions

Mikael Svahnberg*

2024-04-17

## 1 Introduction

This document describes the graded assignments in the course PA1489 Basic Software Engineering.

The course consist of the following assignments:

| Assignment | Size | Description |
|---|---|---|
| Written Assignment 1 | 2.5 hp | Collaboration and Configuration Management |
| Written Assignment 2 | 2.5 hp | Implementation and Documentation |
| Written Assignment 3 | 2.5 hp | Testing and Debugging |

All the assignments span the entire duration of the course, and you need to work in parallel with them (the assignments help and support each other). *Read this entire document first* and plan how you are going to work on all three written assignments during the entirety of the project.

## 2 Assignment at a glance: BurgerOrderer

You and your colleagues have been contacted by a company that wants a system to order hamburgers. They want a web client where customers can roder and tailor their hamburgers, and they also want a second client where the kitchen can see which hamburgers are currently on order and need to be cooked.

With joy in your heart and a song on your lips you accept the mission. This is going to be fun, and you do not care that they can only pay you in small potatoes.

This is when the company adds further requirements. They want you to collaborate well and with the use of modern configuration management tools. They have heard of "git" and want you to use this. They want the different clients to be developed as separate units ("word on the grapevine is that containers is the latest big thing, and we don't mean the kind that you can load on ships!"). Further, they want the project to be of high quality. They have decided that "high quality" consists of three parts. First, everything shall be well documented. Second, there shall be automated testing of key functionality in the project. Third, all developers must demonstrate basic skills such as using a debugger

*Mikael.Svahnberg@bth.se

and a mindset of continually reflecting on what they do in order to enhance their skills.

Ho hum. At least the small potatoes are deep fried. You decide to divide the company's expectations into three components, (1) Show the ability to collaborate with the help of configuration mangement tools, (2) implementation and documentation of the project, and (3) testing and demonstrate the ability to use a debugger. During all three components, you work actively to document what you do in an engineering diary, and when you are done you summarise and reflect upon each of the three components.

Said and done; up, up, and away!

# 3 Written Assignment 1: Collaboration and Configuration Management

Your first task is to decide which development environment to use, and set up your configuration management system. The company wants you to use git, but you are free to decide on your own which server (e.g. *codeberg* or *github*) to use.

## 3.1 Get Started

1. Form a development deam
   - Plan how you wish to work
2. Decide on a development environment
   - It is not a requirement to use the same development environment, but it may help.
3. Decide which git-server to use
   - Make sure everyone has accounts on this server.
4. Create the project. Make sure it has a good name, and describe it properly.
5. In a directory `Planning`, you can start listing the different types of goods (menus, hamburgers, conditments, drinks, etc.) to sell.
   - Use this opportunity to get in your first commit here.
   - Collaborate e.g. by each of you taking responsibility for one type of goods (hamburger, conditment, ... ) to sell.
   - Write in your own text file for that goods type.
   - Swap files and continue
   - Repeat until you have considered all types of goods.
6. In the directory `Reflections`, create a text file for each of you (with your name in the filename) where you document what you have learnt.
   - This is your *Engineering Diary*
   - Which git commands do you use? What do they do?

- What challenges to you encounter? How do you solve them?
- Are there anything that you don't know how to solve? How can you find out more?

With this foundation, you are ready to continue with the other assignments, but do note that you also have to continue working with the items *As the Work Progresses*, and *Summarise and Reflect* at the same time as you work on written assignments 2 and 3.

## 3.2 As the Work Progresses

As you work on the other assignments, the following applies:

1. During the entire development you shall *regularly* update with *commits* as soon as you have implemented anything.
   - Every team member need to actively and regularly contribute with relevant commits.
   - Remember that commit messages shall clearly describe the intent and contents of the commit.
   - Extra points if the work is organised such that development is done in separate branches.
     - Don't forget to document your plan so that you can refer to it in your reflections.
   - It is a bonus, but does not influence the grade, if you handle conflicts where different versions of a file need to be merged.
2. During the entire development you shall *update your engineering diary* with what you have learnt about configuration management.
   - Try to get into the habit of often reflecting upon what you have done, what you have learnt, and what you must find out.
   - It is better to write a little every day rather than a lot once a week
   - Don't forget to commit what you have written so that it is visible in the project log
   - Extra points if you read and summarise external sources on configuration management.

## 3.3 Summarise and Reflect

Collectively write a summarising text in the directory `Reflections`. The following shall be included:

1. Name of everyone in the team.
2. Link to the project's homepage on the git server
3. Short summary about what configuration management is and why it is used.

4. Short summary on the most common workflow with git, including the git commands used.

5. Your experiences of working with configuration management

   - What went well?
   - What did not go well?
   - How did you solve your challenges? What could you have done differently?
   - What did you not manage to solve? Why not?

Copy the text and submit it on Canvas for assessment. Please see further instructions on Canvas about submission dates etc.

### 3.4 Assessment

The follwing is part of the assessment, and is weighed together to a grade on the assignment:

**Regular Contributions** (individual assessment) actively contributes with commits to at least one part of the project

**Diary Reflections** (individual assessment) maintains regular and relevant notes and reflections about working with configuration management tools.

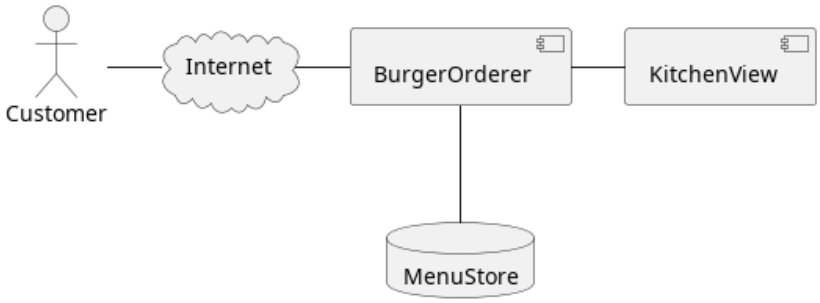
**Well Described Commits** Almost all commits have well formed commit messages that describe the content and intent of the commit.

**Branches** The team actively works with branches in their development.

**Reflective Practice** The work is well summarised with reflections about improvement opportunities.

## 4 Written Assignment 2: Implementation and Documentation

The company has provided an overall architecture for you to adhere to:




**BurgerOrderer** The main web interface

- presents the different types of goods
- The customer can select what to include in their order
- The customer can tailor their order (e.g. remove "onions" from the "Metric Ton Bacon Burger")
- Collects information about the different goods types from the database `MenuStore`
- Once the order is complete, sents via a REST call to `KitchenView`
- *The company sends their regards* and says that you do not ned to make it pretty. Functionality is currently more important than Form.

**MenuStore** a database that contains information about each type of goods.

- Information about the different goods types and how they may be tailored can be managed via a separate interface such as *adminer*.
- *The company sends their regards* and says that you can decide on your own whether to use a relational database or a NoSQL database.

**KitchenView** Receives orders from BurgerOrderer and displays them to the kitchen staff.

- When an order is received via a REST API, it shall be printed on screen.
- *The company sends their regards* and says that it is ok with a text based printout.
- *The company sends their regards* and says that it is also ok if you are not able to interact anymore with the orders (e.g. it is not necessary for the kitchen staff to be able to mark an order as ready for delivery).

The company wishes a container based platform, with separate containers for `BurgerOrderer`, `KitchenView`, and `MenuStore`. You may choose programming language yourselves.

## 4.1 Get Started

1. In the directory `Containers`, create a directory for `BurgerOrderer` and one for `KitchenView`
2. Create a `Dockerfile` for each container
3. Create a `package.json` (or whatever is required by your programming language) and enter any relevant information.
4. Document and/or automate how to build containers and how to run the project.

Plan your work:

1. What is the design of your `MenuStore` database? What information must be there for every type of goods?
2. What is the program design of your `KitchenView`?

3. What is the program design of your `BurgerOrderer`? For example, you may wish separate modules for
    - Web interface and the API endpoints
    - Connection to `MenuStore`
    - Connection to `KitchenView`
    - Formatting the presentation for different types of goods.
    - Handling when the customer orders different types of goods so that they may tailor their order.
    - Summary of what is already added to the order (with the ability to remove parts of the order)
    - Remember that many customers may wish to order at the same time, so have a look at e.g. `cookie-session` to store data while the order is being made.
4. Who is responsible for which part of the project?
5. When should the different parts be complete?

## 4.2 As the Work Progresses

1. *Stick to your plan*
    - It is ok to deviate from the plan if you see that it is necessary, but it should be a *conscious* decision.
    - The earlier you know that you are not able to keep the plan, the more time you have to do something about it.
2. *Document* your work
    - Summarise what every module does.
    - Describe in code comments what every method does.
3. *Commit regularly* when you have implemented something.
4. *Update your engineering diary* with what you have learnt about implementation and developing in containers.
    - Try to get into the habit of often reflecting upon what you have done, what you have learnt, and what you must find out.
    - It is better to write a little every day rather than a lot once a week
    - Don't forget to commit what you have written so that it is visible in the project log

## 4.3 Summarise and Reflect

Collectively write a summarising text in the directory `Reflections`. The following shall be included:

1. Name of everyone in the team.
2. Link to the project's homepage on the git server
3. Short summary of what you have implemented. Describe with 5–10 sentences what you have done and your thoughts on
   - The project as a whole
   - Each container
   - Each module
4. Your experiences of conducting the project.
   - What went well?
   - What did not go well?
   - How did you solve your challenges? What could you have done differently?
   - What did you not manage to solve? Why not?
5. Your experiences of working with comntainers.
   - What went well?
   - What did not go well?
   - How did you solve your challenges? What could you have done differently?
   - What did you not manage to solve? Why not?

Copy the text and submit it on Canvas for assessment. Please see further instructions on Canvas about submission dates etc.

## 4.4 Assessment

The follwing is part of the assessment, and is weighed together to a grade on the assignment:

**Documented Code** All containers are documented. Almost all methods are documented.

**Documented Startup** It is well documented or automated how to run the project.

**Implemented Functionality** The following is implemented (in increasing order of difficulty):

1. *List Types of Goods* The customer can see all different types of goods, as retreived from the `MenuStore` database.

2. *Searchable Database* `MenuStore` contains information about the different types of goods and is being used by `BurgerOrderer`
3. *Order Items* Customers can order items and they are sent to `KitchenView`
4. *List Orders* `KitchenView` receives the orders and prints them.
5. *Adjust Order* The customer can remove items from their order before it is sent to `KitchenView`
6. *Tailor Items* The customer can tailor items on their order before it is sent to `KitchenView`

**Reflective Practice** The work is well summarised with reflections about improvement opportunities in the collective report as well as in the individual engineering diaries.

# 5 Written Assignment 3: Testing and Debugging

The company, finally, expects the project and product to maintain high quality. In this assignment you focus primarily on two criteria for this:

- There shall be automated testing of key functionality in the project, and
- all developers must demonstrate basic skills on using a debugger.

## 5.1 Get Started

1. Plan which parts of the project that should be tested
   - Which modules? Which methods? Which API endpoints?
   - How shall they be tested? How shall they be called? What answers will you get?
   - Which technologies (e.g. test frameworks) do you need?
   - How often shall the tests be run? What happens if a test fails?
2. Make sure that everything that needs to be installed (e.g. test frameworks) are available in each of the containers you are developing.
3. Make sure that it is possible to run the automated tests. Document/automate how to run the tests.

## 5.2 As the Work Progresses

1. *Follow your test plan*
   - Run the tests when you have planned to, and act on the results according to plan.
2. *Document* your work.
   - Summarise what shall be tested, how, and when.
   - Describe your tests (You do not have to describe each individual test, summaries across several tests are sufficient)

- Document the results of your tests.

3. *Commit regularly* when you have implemented something. Test code shall also be configuration management.
4. *Update your engineering diary* with what you have learnt about automated testing.
   - Try to get into the habit of often reflecting upon what you have done, what you have learnt, and what you must find out.
   - It is better to write a little every day rather than a lot once a week
   - Don't forget to commit what you have written so that it is visible in the project log

## 5.3 Conduct and Document a debug session

Some time during the project, each of you shall conduct a debug session and document this in your engineering diary.

1. Select some functionality, e.g. *Order a "Dripping With Lard Heartstopper" menu*
2. What breakpoints do you set in order to start the debug session? Where do you find the files to set the breakpoints in?
3. How do you continue? How do you use the buttons for "Continue", "Step Over, "Step Into", and "Step out"?
4. Watch some variable
   - How do you do this?
   - What is its current value?
   - Can you find out when the value changes? How?
5. Try some different "paths" through the functionality, e.g. order something else, cancel half-way through, etc.
   - How does this effect which code is being executed?
   - How does this effect your watched variables?

Every step shall be documented. Repeat every step so that you are confident on how it works.

Finalise by summarising and reflecting, for example:

- What went well? What did not go well?
- What was easy? What was difficult?
- Can debugging become a useful tool for you? Why? Why not?

1. Tip Debugging a node.js application inside a container is not quite the same thing as debugging something developed locally. One guide to follow for vscode/vscodium is `https://medium.com/fl0-engineering/the-best-way-to-debug-a-node-js`

   Suitable search terms to find how to do this for other prohgramming languages is e.g. *"debug <program language> in container"* ,possibly you may also wish to add *<development environment>*.

## 5.4 Summarise and Reflect

Collectively write a summarising text in the directory `Reflections`. The following shall be included:

1. Name of everyone in the team.
2. Link to the project's homepage on the git server
3. Short summary of the functionality you have tested.
4. Short summary of how you conducted your tests.
5. Printout from your most recent test session so that you can see
   - How many tests you have written
   - What is being tested
   - How many tests that pass and how many fail.
6. Your experiences of writing automted unit tests.
   - What went well?
   - What did not go well?
   - How did you solve your challenges? What could you have done differently?
   - What did you not manage to solve? Why not?
7. Link to the documentation from your debug sessions in each of your individual engineering diaries.

Copy the text and submit it on Canvas for assessment. Please see further instructions on Canvas about submission dates etc.

## 5.5 Assessment

The follwing is part of the assessment, and is weighed together to a grade on the assignment:

**Test Plan** It is documented what to test, how to test, how often, and what to do if any test fails.

**Tested Functionality** Unit tests have been written that test some part of some functionality in the system.

**Working Unit Tests** The unit tests can successfully be run by following a description from the test plan.

**Follows the Test Plan** The tested functionality follows the test pöan.

**Documented Experience of Debugging** (individual assessment) documentation and reflections from a conducted debug session.

**Reflective Practice** The work is well summarised with reflections about improvement opportunities in the collective report as well as in the individual engineering diaries.

# 6 Conclusion

"Any Questions?"

You look at the company representative and quietly hope that they do not have any questions to ask. You have just completed a fantastic presentation that desfribes in detail how your BurgerOrderer system works and how it uses all the Recent Technologies™ with `git`, containers, and unit testing. You have shown how everyone in the project of course knows how to use their development enviromment and that you actively and naturally use your engineering diary in order to reflect on your work and see how you may improve.

You hope that the company representative will not ask too many questions about how you could miss something so basic in your tests, and that little mishap with several concurrent customers, but you do believe that you were able to divert attention away from these – after all - minor deficiencies in your project. Actually, if you ignore the flaws, it really was very nearly perfect.

The company representative leans forward and looks down in their notebook. They clear their voice and ask:

"Are you sure that you have pushed all the commtis and uploaded all three reports on Canvas?"

... which of course you have.