SOFTWARE ENGINEERING

Exercises

Summary
This document contains the exercises that support the lecture Software Engineering.

Prof. Dr. habil. Florian Kellner
florian.kellner@th-rosenheim.de

Software Engineering

Contents

1 Exercise 1: Good preparation is everything!

*"In a mature software development organization, much of a software engineer's life is spent in meetings, discussing **requirements**, planning, evaluating software products [...], **documenting** and **reporting**."*
[Conn, R. in IEEE Software, No. 5, 2002]

1.1 Task 1: Software? Processes?

By now you have learned a bit about process models.

a) In which phases of the software life cycle is software created?

b) Discuss the three basic models "Sequential/Waterfall," "Sashimi," and "Incremental" regarding the required competences of the development team, the scope of projects and the requirements for projects. What do they have in common, how do they differ?

1.2 Task 2: Tools check

In addition to your head, paper and pencil, you will also need and use a few tools for the following exercises, namely:

- Word processing: MS Word, alternatively: LibreOffice Writer, Google Docs
- Flow charts: MS Visio, alternatively: MS PowerPoint, Google Slides, LibreOffice Impress, **Miro**, Collaboard, **draw.io**, or any other drawing program
- List creation, diagrams: MS-Excel, alternatively: LibreOffice Calc, Google Sheets
- Process models: V-model project assistant 1.5.8[1]
- Web Browser, Email: Mozilla Firefox, Google Chrome, MS Edge, Outlook, Thunderbird, or any other browser and email client
- Configuration Management Client: TortoiseGIT, alternatively: SourceTree, **GitHub Desktop**
- Mind Maps: Freeplane (FreeMind), alternative: MS Visio, **Miro**, Collaboard
- UML diagrams: MS-Visio, alternatively: ArgoUML, UMLet, plantuml in GitLab, Enterprise Architect, **draw.io**, ...
- Creation of presentations and drawings: MS PowerPoint, alternatively: LibreOffice Impress, Google Slides

a) Check on your computer whether the software mentioned is installed and whether you can run it.[2]

b) Preferably use your own computer. If necessary, install the specified software there.

---

[1] Portable version available e.g., at
https://www.cio.bund.de/Web/DE/ Architekturen -und-Standards/V-Modell-XT/vmodell_xt_node.html .
[2] On the laboratory computers: virtual desktop via https://inf-view.fh-rosenheim.de/ .

1.3   Task 3: Word template

In the project business and in your studies, a word processor is one of your most elementary tools. Using a Word template as an example, you should learn how to use the most important functions of MS Word:

- Structured work with a word processor (Word)
- Creating a Word document template
- Creation of / working with format templates (e.g., chapter, section, ...)
- Indexes: Generate table of contents, outline view, ...
- Manage links: Number figures, code fragments
- Essential tools: Ctrl-C, Ctrl-V, Ctrl-X, Ctrl-Z

a)   Create a Word document as a template – for example – for a report. The document should have the following properties:

- Cover sheet (title, student)
- Table of contents (chapters with page numbers)
- Numbered pages in the footer (no numbering on the cover sheet!)[3]
- The following numbered main chapters:

  1. **Summary**
  2. **Tools**
  3. **Remarks**

b)   Working with the template

What do you think, what is software and what does software engineering mean? To do this, write a few sentences under Chapter **1. Summary** and insert any image (gif, jpg, png, ...) there. Create a caption for this figure, including a figure number.
Describe the figure with a short text, in which you refer to the caption. The referencing should be done in such a way that the reference is automatically adjusted if something changes in the figure (number, caption).

c)   Extension of the template

On the cover sheet of your report, add a line that indicates where the document is located in the directory structure. The field should be updated automatically.
In addition, add the current date to the footer, this should also be updated automatically.

d)   *Optional (try it!): Advanced extension of the template*

In addition, on all pages – except for the cover sheet – add a field with the title that is on the cover sheet in the header.
*The field should be automatically updated if the title on the cover sheet changes.[4]*

---

[3] Note: see also "Different first page" or "Section breaks".
[4] Note: see Create "Bookmark" (on cover page) and "Cross-reference →bookmark" (for header).

1.4   Task 4: Try it out and interact

Use the Word template you created in Task 2 and create a sub-chapter under Chapter **2. Tools for each of the following sub-tasks**.

a)   Create the following class diagram with **draw.io** or MS Visio[5]
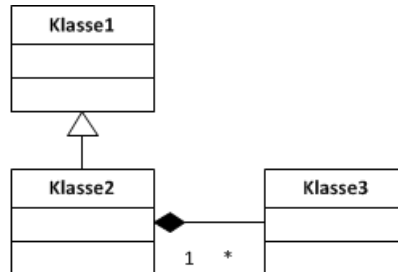
and paste it into the Word document.

b)   Start the V-model project assistant and insert a screenshot of the initial dialog window into the Word document.

c)   Take a screenshot of TortoiseGIT's (or SourceTree's, **GitHub Desktop**'s) About dialog and paste it into the Word document.

d)   Finally, create the following mind map with Freeplane, **Miro**, or Collaboard.
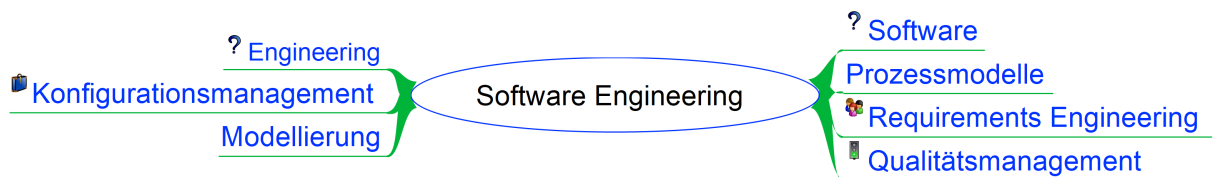
and paste it into the Word document.

---

[5] Template Category = "Software and Database," Template = "UML Class"

2 Exercise 2: Procedure in software projects

The aim of this exercise is for you to use a small example to understand the steps involved in software development projects. You should be able to think of a simple procedure yourself and model it as a flowchart. And you know how to represent roles, products/results, and activities.

Finally, you will get to know the main factors influencing projects and examine them using the following example.

Story

Congratulations! You have just founded a new Rosenheim software and consulting company (RoSCoF) and are now producing web applications for your customers.

Your first customers are Silvio Bocaccio[6] and Umberto Rossi, both owners of the pizzeria "Il Pappagallo." Your two customers want you to build a web application that can be used to order pizza and other foods over the Internet; so, they want a delivery service software. Both customers are available to you on site for a few hours a day to tell you about the requirements for the web application.

You decide not to write a full specification (you don't know what that is yet)! Your customers should write the requirements for the functions/features of the web application on *index cards.* A file card contains no more than three sentences of what a required function should look like.

Example of an index card:

---

[6] Names of the customers and the pizzeria are fictitious. Matches with living persons and/or premises would be purely coincidental.

2.1   Task 1: Creation of a process model

Your task as the manager of RoSCoF is now to think of a *process (model) for your five ambitious employees* with which the web application can be programmed.

a)   Think about how the requirements on index cards are implemented in a delivered partial web application.

The following aspects must be considered in your approach:
- Programming the requirements of index cards, of course only if feasible.
- Developer tests the requirements of flashcards, and develop a unit test per requirement (preferably even before implementation → "Test First")
- Delivery of the software to Silvio and Umberto.
- Acceptance test of the requirements of individual index cards by the customer.
- To be able to work in a focused manner, only the project manager is allowed to have customer contact.

b)   Requirements for the procedure.

Define:
- *Roles* for the five employees (not Mr. Müller, but: "Developer", ...), the role "Customer" is also included.
- *Activities* in development (e.g., writing source code, testing software, ...).
- *Intermediate results* (e.g., source text).
- A *flow* that shows the order in which activities are performed, what intermediate result they contribute to, and who is responsible for the activities.

c)   Finally, document your process, i.e., also roles, activities, and intermediate results, as a flow chart using MS Visio, **draw.io**, by hand or with any other tool you prefer 😊 .

For example, start like this:
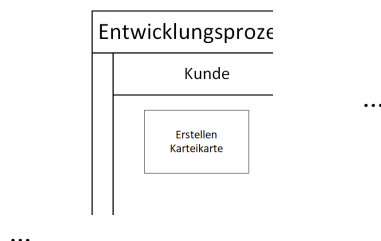
In the appendix you will find flow chart symbols (according to DIN 66001) and an example with so-called "swim lanes" to help you.

2.2   Task 2: Factors influencing projects

As the manager of RoSCoF, you also want to be able to specify a suitable strategy. It helps to classify your development project for the delivery service software.

Boehm and Turner, in their book "Balancing Agility and Discipline" proposed a classification scheme for projects. The scheme has five criteria:

Criteria	Description
Personnel	How well qualified are the people in the project? What is the proportion of IT professionals (Cockburn Level 2 and 3)[7]? Possible levels here are 35%, 30%, 25%, 20%, 15%
Dynamism	What is the proportion of requirements that change monthly? Possible levels here are 50%, 30%, 10%, 5%, 1%
Culture	How large is the proportion of people who want as much freedom as possible when designing their projects? Possible levels are here: 90%, 70%, 50%, 30%, 10%
Size	How many people work on the project? Possible levels are: 5, 10, 50, 100, 500, ...
Criticality	How (safety) critical is the project? Possible levels here are: Loss of comfort, loss of (disposable) funds, loss of essential funds (=company bankrupt), loss of a life, loss of many lives

a)   For the project outlined in Task 1, draw a Kiviat diagram.

(Find out what a Kiviat diagram is first!)
If necessary, make suitable assumptions as to which stages might be suitable for your project and <u>document </u>them.

b)   *(For advanced)*

Create an **Excel sheet** *in which you can specify the 5 criteria for any project – preferably using a drop-down list[8]– and a corresponding Kiviat diagram will be displayed automatically!*

---
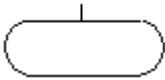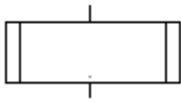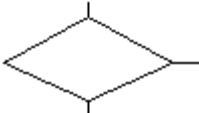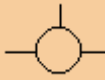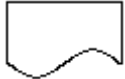
[7] Cockburn means by level 2 and 3:

Levels	Characteristics
3	Able to revise a method (break its rules) to fit an unprecedented new situation.
2	Able to tailor a method to fit a precedented new situation.
1A	With training, able to perform discretionary method steps (e.g., sizing stories to fit increments, composing patterns, compound refactoring, complex COTS integration). With experience can become Level 2.
1B	With training, able to perform procedural method steps (e.g., coding a simple method, simple refactoring, following coding standards and CM procedures, running tests). With experience can master some Level 1A skills.
-1	May have technical skills, but unable or unwilling to collaborate or follow shared methods.

[8] In Excel: Data → Data Tools → Data Validation… →Allow: List

Appendix: Flow chart symbols according to DIN 66001

Flowcharts are widely used in computer science. They are used, for example, to document program sequences and algorithms (program flow charts). They can also be used to model the procedure in software development projects or business processes and other processes in general.

Symbol	Meaning according to DIN 66001	Meaning in SE
	Border post (terminator)	A terminator is the **start and stop symbol** of a process model. This is where the process ends: e.g., software delivered, project canceled, ...
	Processing in general (process)	With the "Process" symbol, **activities are** modeled, e.g., program class, test class ...
	Subprogram call (predefined processes); Reference to documentation elsewhere in the form of clear internal labeling	Using "predefined process" large flowcharts are structured. The symbol indicates the call of a **sub-process (sub-procedure).**
	Flow lines	The **control flow** / sequence from processing step to processing step is shown with the aid of the flow lines.
	Branch (decision)	The branch symbol represents **decisions** where the control flow can branch in different directions, e.g., found a bug? Yes / No, build successful? Yes / No, …
	Connection point (connector)	Various possibly **parallel control flows are** brought together again or parallel control flows are generated via a connector.
	Data on Document	**Results** / **products** / **artefacts are displayed** with the "Data on Document" symbol: e.g., specification document, a Java class, ...
	Data on memory also with direct access	This symbol represents **access to stored data.** This can be the messages in the bug tracker, the source code in the repository, and others.
	Input/output; also: data, general (data)	

The symbols from DIN 66001 cannot represent who is carrying out an activity or who is responsible for a result (role). For this we use the "Swimlane" notation, which you will also get to know in the context of the UML activity diagrams. For example, MS-Visio offers "**Cross-Functional Flowchart Shapes**".

Example of a flowchart with swimlanes[9]

---
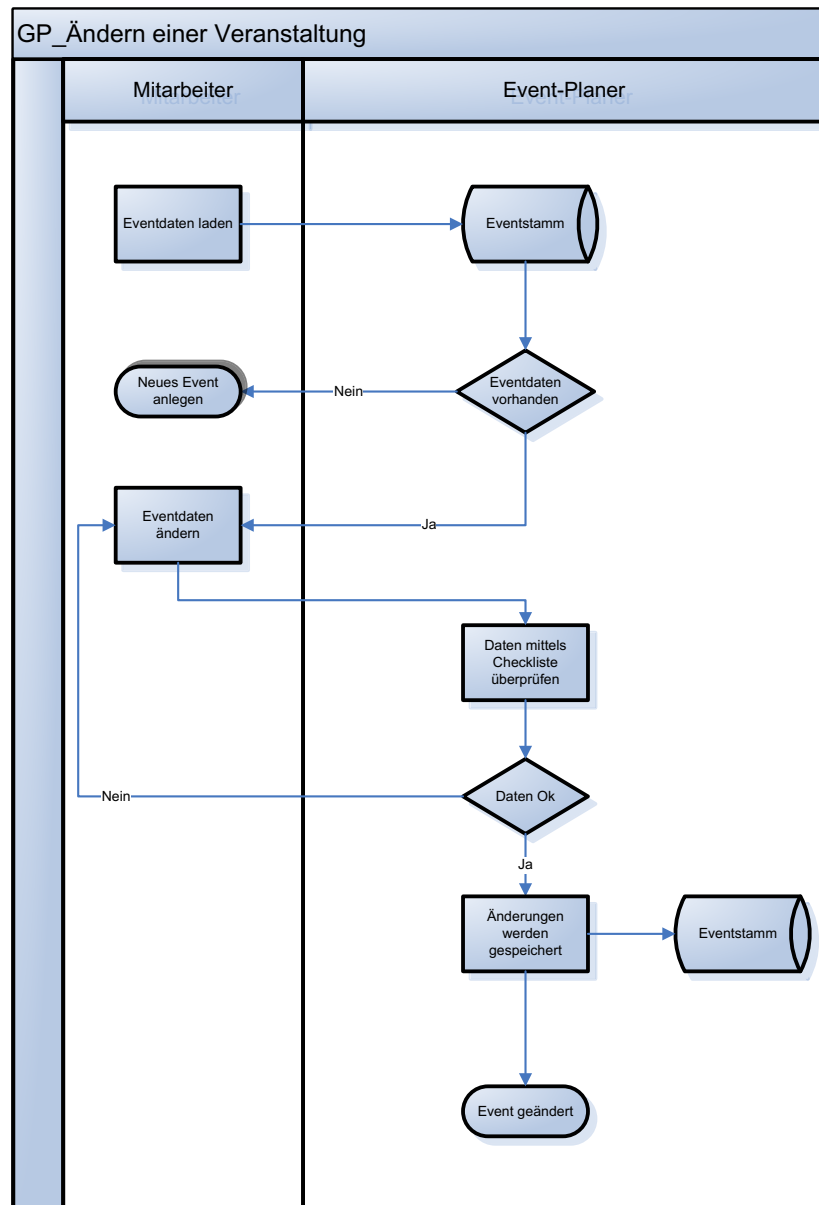
[9] Source: SE-2 project "Event Planner", S. Keller et al.

3 Exercise 3: Process models

Assessing the suitability of a process model, but also being able to implement a given process model in a specific project, characterizes a good software engineer. In today's exercise, you will learn about both.

3.1 Task 1: Selection of a process model

Which traditional model would you choose for the development of the following software systems?

a)   A software system that controls an anti-lock braking system in a car.

b)   An accounting system that replaces the existing system in a company.

c)   An interactive system for train passengers to find train departure times at train stations.

d)   A virtual reality system for power plants. The 3D software to be developed should show what happens inside power plants. With its help, it should be possible to simulate the processes inside a power plant and to display them as a three-dimensional graphic. In the future, it should be easier to construct and operate power generation plants.

Think of plausible criteria for each of these software systems that could apply to their development, and use them to justify your choice!

3.2 Task 2: V-Modell XT project assistant

The management of the company RoSCoF (cf. Exercise 2) introduced the V-Modell XT as a standard process model. As a project manager, you are now supposed to be in charge of a software development project. As expected, it is about the development of the said pizza delivery service software.

The following framework conditions are given:

- The project is scheduled to start on **April 5, 2022** and be completed before Christmas (**December 22, 2022**).
- As part of the development, you will first create **two prototypes**: One prototype should implement the user interface and thus facilitate coordination with your customer (duration approx. 3 weeks). The second prototype will be a technical breakthrough (duration approx. 2 weeks).
- After completion of the prototype development, the development should be carried out incrementally. There should be **two increments**, with the 1st increment being delivered at the beginning of September and the 2nd increment being delivered in the second week of December.
- Since the resources are already planned, you don't need to schedule more than 1-2 days for project definition and project completion. Your team includes you as project manager and 4 software developers.
- You have been instructed to use off-the-shelf products (third-party software, frameworks, etc.) in the project, provided they are available and economically viable.
- To create a project plan, you build on your experience from other projects. You need about 1 week to prepare an offer, and it usually takes about 3 weeks until the order is placed. You plan about **10 weeks for each increment** (creation of specifications about 3 weeks, rough and fine draft about 3 weeks, implementation about 4 weeks).[10]
- No milestone (decision point) falls on a Saturday, Sunday, or public holiday.

Since you have the project order, you must now set up the project using V-Modell XT and its tools:

a) Start a new project and initialize it using the V-Modell XT project assistant[11].

b) Complete the tailoring with the project assistant.
   Briefly explain each case *if* you do *not select* certain modules.

c) Create a milestone plan based on the data you have.[12]

Start:	Projekt genehmigt 05.04.22	...	End:	Projekt abgeschlossen 22.12.22
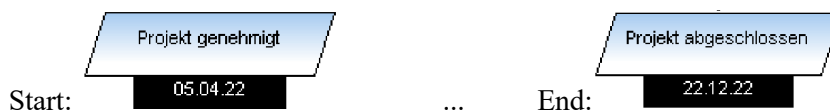
d) Export the project plan in Microsoft Project (XML) format and view the corresponding Gantt chart.[13]

---

[10] The decision points for specifications, rough draft and detailed draft are identified in the V-Modell XT documentation (see also: *Reference processes) with:*
 • "**Overall system designed**" (Requirement specification = functional specification is available)
 • "**System designed**"   (System architecture = rough design = logical architecture is available)
 • "**Unit(s) designed**"   (SW or HW architecture = detailed design = technical architecture is available).
[11] Depending on the installation, a *source directory for the V-Modell XT may first have* to be specified (.xsd files), e.g., `C:\...\Release-2.3\V-Modell XT\Modell`.
[12] Insert so-called "free milestones" for the creation of the two prototypes.
[13] It should also work with ProjectLibre or Planner.

4 Exercise 4: Configuration management

Efficient provision of essential and always up-to-date project information as well as configuration management are decisive factors for successful development projects – right from the start!

Configuration management is one of the central tasks in software engineering. Only then is sensible software development in a team possible. Two of the best-known free tools are **Git** and **SVN** (Subversion). With these tools it is possible to manage code (i.e., ASCII texts) and binary files (e.g., Word documents) professionally. As part of this exercise, we now deal with basic *Git* commands.

Wiki pages are a simple and effective means of displaying current project information. Again, there are a wide variety of integrated variants (*Redmine*, *GitLab*, *trac*, etc.). In this exercise, we'll look at *GitHub*.

However, a project first needs a so-called repository...

4.1 Task 1: preliminary work: teams and project repositories

a)   Form teams of no more than **five** people and designate a team administrator for the project repository.

b)   The team administrator creates a project on GitHub[14].
- Create a user account on GitHub and login.
- On the main page, click the **New** button (Create blank project) and enter a "***Repository name***" for the team as follows.

     Repository name: **thro-se**

- Enter an appropriate description for the project and select ***Private*** as ***the Visibility Level***. The project created in this way is displayed on the overview page.

4.2 Task 2: Team Administration

After creating the project, the project owner (team admin) now needs to add the team members. In order for this to be possible, they ***must have registered at GitHub***.

When this is done, in the project under ***Settings*** ➔ ***Collaborators*** (left side menu) Team members are added with **Add people**.
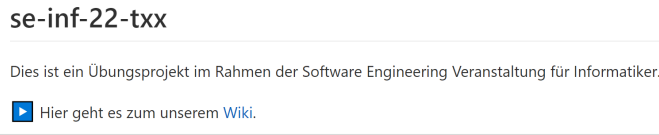
Check (every team member!) if you have access to your project. Your email inbox should also contain a notification regarding your project affiliation.

---

[14] https://github.com/

4.3 Task 3: Creation of a project homepage (wiki)

Create your project appearance together in your team. All relevant project information is typically stored there in order to make it easy to find for those involved in the project. To do this, a README file and a wiki are created in a GitHub project.

a) Create a README.md for your project, something like this:

> ### se-inf-22-txx
>
> Dies ist ein Übungsprojekt im Rahmen der Software Engineering Veranstaltung für Informatiker.
>
> ▶ Hier geht es zum unserem Wiki.

Notes on syntax can be found here[15].

b) Now create the wiki page for your project (Wiki). Briefly repeat the content of your README here and add a description of your team to the page. Present your team there in the form of a table, with pictures if you like, but at least with the rows (or columns):
- Name
- Role   (project manager, developer, ...), and
- Contact   (= email address).

c) Now add the heading "**Our domain**" to the page in such a way that clicking on it takes you to a separate wiki page.

4.4 Task 4: Cloning the repository

Everyone on your team works with a complete project repository, meaning they have all versions of all files in the project in a local directory. The content of the local directory is compared with the central (remote) repository via Git. Now you'll learn some basic configuration management concepts and apply them with Git.

a) When the project was created, a repository was automatically created on the Git server. In order to now synchronize your data with the repository, you need a suitable tool.

In the following, the use with *GitHub Desktop* is described, but the use of other tools such as *TortoiseGit*, *SourceTree*, or the command line is possible without further ado (a list of alternatives can be found here[16], a tutorial for using the command line tool can be found here[17]).

GitHub Desktop can be downloaded from the following website: https://desktop.github.com

---

[15] https://inf-git.fh-rosenheim.de/help/user/markdown.md
[16] https://git-scm.com/downloads/guis
[17] https://try.github.io

b) Any action with the Git server requires you to be authenticated. The fastest way is authentication via username and password. To do this, select in *GitHub Desktop* "***Clone a repository from the Internet***" and add the "**https:**" URL specified in the GitHub project, for example:

> https://github.com/kef59000/thro-se.git

You can now create files in the directory you have selected (destination path) and save them in the central repository.

> ***Note:*** *Difference between **Clone** and **Pull***
> - *Cloning is used to initialize your workspace / local repository.*
> - *Pull only fetches changes from the remote repository.*

(***Not within the scope of this exercise,*** *but...) In order to avoid the tedious task of entering user names and passwords, you could also authenticate using an SSH key if you are interested.[18] With the Git Clone dialog, you would then have to enter the "SSH" URL, for example* `git@github.com:kef59000/thro-se.git.`

4.5 Task 5: Make files available to the team

a) A team member now creates a subdirectory `src` in the project directory and a file in it, named `.gitkeep`[19].

> ***Caution:*** *The Gitkeep file is only required to create folder structures and can be removed again as soon as the folders are filled with data.*

b) A `HelloWorld.txt` file should be created in the root directory. Once created, this file can be made available to other team members. To do this, select "***Commit to main***" in *GitHub Desktop*. Before, check the checkboxes next to the directory and the file and write a comment (Commit Message / Summary).

> *It's good form to include a **meaningful comment** with **each commit**—here, for example, "Initial build."*

With Commit the changes are transferred to the local repository.

c) In order to transfer these changes to the remote repository and thus enable the team members to access them, a **push** must be carried out ("***Push...***").

d) All team members now clone the project, if they have not already done so (task 4b), or carry out a **pull** ("***Pull...***").

The `HelloWorld.txt` file should appear along with the `src` folder and the `.gitkeep` file to be visible to all other team members.

---

[18] Instructions, for example, at https://help.itc.rwth-aachen.de/service/ubrf9cmzd17m/article/54a96b4470644bf9a015da5b197da489

[19] http://stackoverflow.com/questions/7229885/what-are-the-differences-between-gitignore-and-gitkeep/7229996#7229996

4.6   Task 6: Familiarization with how Git works

Next to the `.gitkeep` you can store `.gitignore` files in the repository. This is how Git is told which files to ignore[20]. This can be specified separately for each directory.

a)   In the `src` folder, create a `.gitignore` file with the content `*.tmp`. Now add a file with the extension `.tmp` and watch *GitHub Desktop* behave on your next commit.

b)   Try as you like! Complete the process

   1. Adding not indexed files
   2. Commit
   3. Push

   using the [cheat sheet][21] (alternative [source][22]).

4.7   Task 7: View version history and changes

a)   Make multiple changes to a file in a row. The routine in everyday project work is usually as follows:

   1. A **Commit** follows after logically completed units – with a suitable comment!
      (This can be repeated until a "ticket" has been processed.)

   2. Then a **Pull** is performed to get any changes made by team members.
      (This can lead to conflicts and may have to be merged at this point – "**merge**", see Task 8)

   3. Finally, a **Push** occurs to propagate the local changes to the remote repository.

b)   View the version history, comments, and changes (***History***) and have the differences for two versions of this file displayed (***GitHub.com***).

c)   Rename the `HelloWorld.txt` file to `readme.txt`, check in this file (**Commit**) and then look at the version history of the renamed file.

4.8   Task 8: Conflicts and Merging (merge)

Two team members now modify the new file in parallel (first **Pull** then **Push**) and try to transfer it one after the other.

What happens to the second team member?

a)   Investigate the following cases:

   1. Both team members modify different lines of the file.
   2. Both team members modify the same line of the file.

   Either case leads to a conflict that cannot be easily resolved:
   Both when opening the `readme.txt` file and in *GitHub Desktop*, the conflict is represented by reflecting both team members' changes in the text. The text file might look something like this because of the same changed line:

---

[20] https://git-scm.com/docs/gitignore
[21] http://ndpsoftware.com/git-cheatsheet.html
[22] https://git-scm.com/book/en/v1/Getting-Started
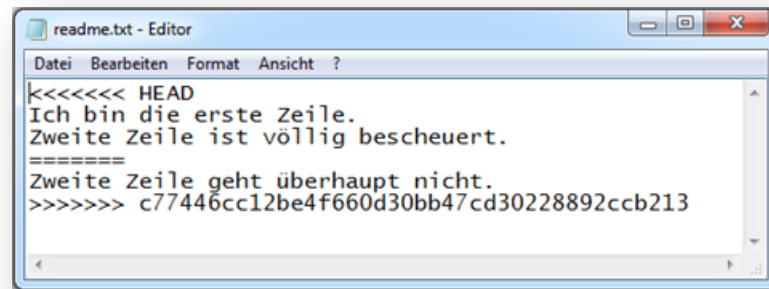
Now edit the file appropriately and mark the conflict as resolved.[23]

(Don't forget to commit afterwards!)

b) Provoke a conflict again! However, when reporting the conflict, do not execute a Pull or correct it manually, but call up the log instead.

(Don't forget to commit afterwards!)

What happens and how is a conflict represented as it progresses?

4.9   Task 9: GitLab tools and interaction with Git

GitLab offers a variety of options for project management and project presentation. Let's just try a few more useful things...

a) Set your project using the **New milestone** button (Menu on the left *Issues* ➔ *Milestones*) the two milestones "*M1 – Setup*" and "*M2 – Development*" to which tasks (**Issues**) can be assigned later.

b) Use the **New label** button to create two labels "*1st sprint*" and "*2nd sprint*" with different colors for your project so that you can later mark tasks (**Issues**) accordingly.

c) Issues can be easily presented and tracked on issue boards.
   • Click on **Boards** and create a new board "*Setup*" pull down menu➔ *Create new board*), with the two lists "*Open*" and "*Closed*".
   • Edit the new setup board (➔ *Edit board)* and select "*M1 – Setup*" for Milestone so that only issues that are assigned to this milestone are displayed there.

d) Here is another simple way to display Product Backlog and Sprint Backlogs: Select the *Development* board (should already be created by default). Configure it in such a way that you insert two more lists (*Create list*) between the two lists "*Open*" and "*Closed*", namely with the two labels "*1st Sprint*" and "*2nd Sprint*".

e) It is often desirable to be able to access relevant file versions directly from the project wiki. This is also possible with GitLab. Try the `readme.txt` file

   • with the current status
   • an older version of the file

to include in your project wiki and to link to Git [24]. To do this, formulate an issue with 2 tasks (*in Write* ➔ ▥) for the *M1* milestone and assign it to one of your team members (➔ *Assignee*).

---

[23] An external conflict resolution tool could also be used for this.

f) Observe the automatically logged activities on this issue, especially when a task has been marked as complete. When both tasks are done, it should be closed (➔ **Close issue**). Watch your **setup** board.

g) With GitLab / Git it is conveniently possible to mark tasks as done with a commit. Close such an assigned task without using GitLab 's graphical interface, but solely with the help of a commit message (see notes[25] in the GitLab Docs).

4.10 Task 10: Cool features

a) Draw models. Try drawing the following simple domain model as a class diagram using *plantuml* directly in your wiki under **Our domain** (Task 3c).
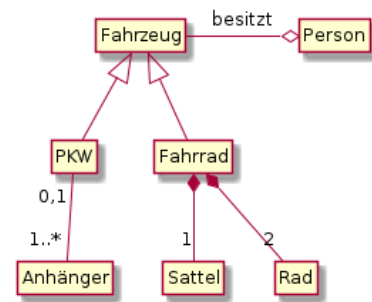
Hints:
- *See https://plantuml.com/de/class-diagram, instead of the tags given there* **@startuml** *and* **@enduml**, *use the* ``` **plantuml** *and* ``` *tags in our Gitlab Wiki*
- *Work with **Preview** before you save.*

b) Parallel development with branches.
- In your repository, create a new development branch called "*dev*" via **Branches** ➔ New branch.
- Switch to the *dev* branch (pull down menu➔ *dev*) and change, for example, the readme.txt so that you add a test line below and then confirm the change with Commit changes.
- Now the changed file should be transferred back to the *main* branch. To do this, create a **Merge request** in the *dev* branch with an assignee and a team member as a reviewer. In addition, specify the milestone "*M2 – Development*" and the label "*1st Sprint*" before you click Create merge request.
- For the merge to complete, the **Merge request** must first be confirmed (Approve) and then triggered with Merge.
- Finally, review the process under **Repository** ➔ *Graph*.

c) Continuous integration and deployment (CI/CD).
You could also use GitLab to trigger automatic actions, such as builds or tests, on every Commit. Basically, this is only done via a file .gitlab-ci.yml with an even simpler markup language " **YAML A** in't **M** arkup **L** anguage". This would actually go beyond the scope here. Therefore, it is only mentioned here for further study... see, for example, *"Hello world" with GitLab CI* (integrated)

---

[24] Tip: In the GitLab project under **Repository** ➔ *Files* it is possible to browse through the repository.
[25] https://docs.gitlab.com/ee/user/project/issues/managing_issues.html#closing-issues-automatically

4.11 Task 11: Parallel development with branches (Branching strategies, Gitflow)

There are different branching strategies to support parallel development:[26]
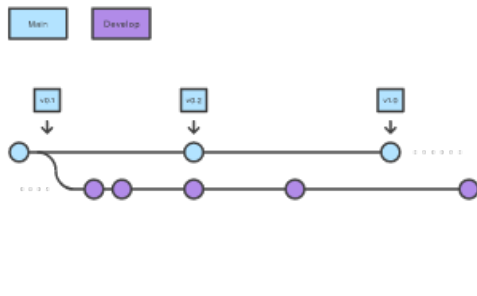

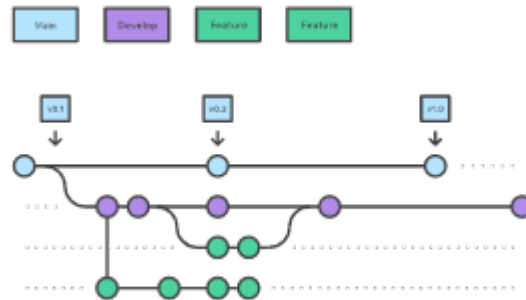*Figure 1. GitFlow: Main & Dev Branches*

Figure 2. GitFlow: Main, Dev & Features Branches
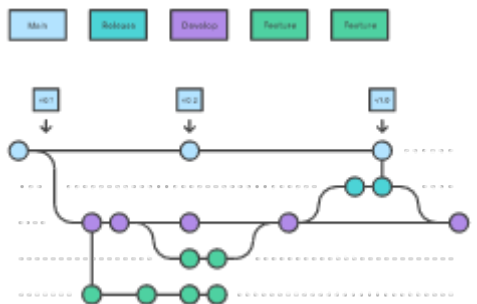
Figure 3. GitFlow: Main, Dev, Feature and Release Branches
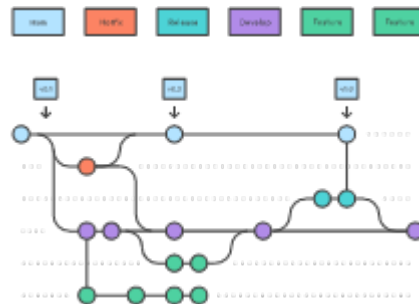
Figure 4. GitFlow: Main, Dev, Feature, Release & HotFix Branches

---

[26] https://www.atlassian.com/de/git/tutorials/comparing-workflows/gitflow-workflow, https://nvie.com/posts/a-successful-git-branching-model
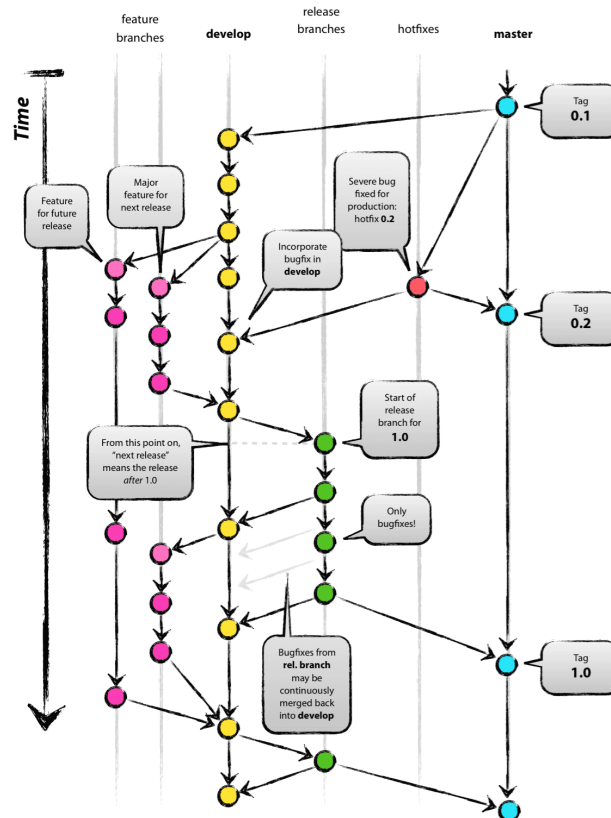
a) In *github.com* or in *GitHub Desktop*, create a new development branch called "dev".

b) Switch to the dev branch and change, for example, the readme.txt so that you add a test line below and then confirm the change with Commit changes.

c) Now the changed file should be transferred back to the main branch. To do this, create a **Pull request** in the dev branch with an assignee and a team member as a reviewer.

d) Finally, delete the dev branch.

4.12 Task 11: Continuous integration and deployment (CI/CD)

You can also use GitHub to trigger automatic actions, such as builds or tests, on every Commit. Basically, this is done via an yml files with an even simpler markup language " YAML A in't M arkup L anguage". This would go beyond the scope here. Therefore, it is only mentioned here for further study... see, for example, https://github.com/features/actions

---

PS: You can delete your project again via the ***Settings*** ➔ Delete this repository.

5   Exercise 5: Requirements Engineer.: Determine Requirements

After you have already become clear about the general procedure for software development in your company RoSCoF (Exercise 2) and have also created a concrete project plan (Exercise 3), we can now get started. So, you are **a contractor,** and you have to create the new pizza delivery service system for the pizzeria.

> First read the text in the **appendix** about the "Bringdienst" carefully!

5.1   Task 1: Extract information

Create a **mind map,** for example, using Miro or by hand on a piece of paper.

Use this mind map to get an overview of all the <u>relevant</u> information for the delivery service system you have commissioned. Gather all the information about your customer that seems important to you (only what can be gleaned from the text in the appendix, so don't invent anything about it).

As a guide, start with the following structure:
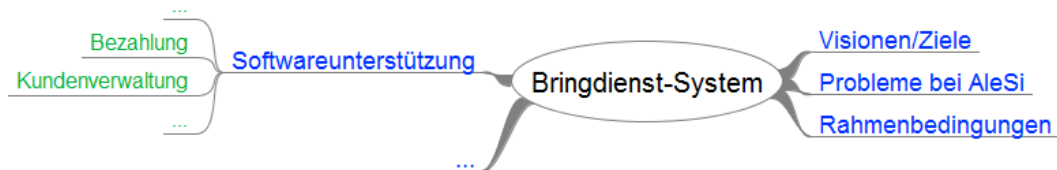
*Tony Buzan, inventor of the Mind mapping technology, has established the following **basic rules:***

1. *Start with a colored image in the center.*
2. *Capitalize all words. This promotes clarity.*
3. *The words should be on lines. Each line should be connected to other lines.*
4. *Use only one keyword per line if possible.*
5. *Where possible, include images and symbols.*
6. *Use as many colors as possible.*
7. *Ignore your controlled thinking. If possible, record everything that comes to mind in connection with the central idea.*

Tony Buzan
* 2.6.1942, † 13.4.2019

5.2   Task 2: Create a stakeholder list, identify interests

Identify <u>all</u> stakeholders of this pizza delivery service system, especially those mentioned in the text (these should also appear in your mind map, for example under general conditions) and who you would add from your own experience.

It is best to create a table (**Excel** sheet) and document **the role** and approximate **interests** of each **stakeholder** (3 columns).

Attachment: Description "Delivery service"

Motivation

Alessandro and Silvio have a small *AleSi* pizza service in Rosenheim. The service delivers pizzas and other dishes, as well as drinks to your home. Alternatively, orders can be picked up directly by the customer. Business is going very well, and the customer base is growing continuously.

Many delivery services already offer orders for food and drinks over the Internet. Alessandro and Silvio expect further sales growth from the possibility of ordering via the Internet and would therefore like to invest in such software.

The delivery service has expanded in recent months and now has a total of three branches in the greater Rosenheim area.

Each branch currently only has an electronic cash register, a telephone system, and no other IT support. Orders are noted on slips of paper, drivers are scheduled on a large board. Bookkeeping is done via a separate BuChi system.

Currently, also because of the expansion, there is chaos in the delivery service of Alessandro and Silvio in Rosenheim:

- Data about orders, customers and inventory is scattered across multiple Excel spreadsheets, Post It's and notes on pin boards.

- Due to the chaos of notes and the information scattered across three branches, accounting for the delivery service has become extremely difficult and time-consuming.

- Personnel planning is currently done using a board. It is important that there are enough drivers available every evening. That often didn't work out and Alessandro had to step in as a driver himself.

- Holgi, the computer expert, has had to import old backups several times since Silvio cannot use the BuChi accounting system and constantly deletes the database with Explorer.

Software is therefore to be created for the AleSi delivery service, with which food and drinks can be ordered over the Internet. In addition, internal ordering should also be made more efficient using software.

Information about the delivery service

Quotations and orders

The delivery service delivers Italian dishes such as pizza and pasta as well as drinks and ice cream. There are often special offers, such as for the European Football Championship, where special additional dishes related to football are offered.

The dishes from the menu can still be modified in the orders: ingredients can be left out; others can be added (e.g., extra olives and pepperoni). Pizzas come in two sizes: single and magnum.

Orders can currently be placed in stores and over the phone. Both owners would like to be able to take orders online.

Telephone orders are currently still being noted manually on a pad, like in old restaurants. An invoice is then created with a cash register system for delivery.

Deliveries are made throughout the city. However, different delivery locations differ in the minimum order quantities or the surcharges if the minimum order quantity is not reached. Customers who also pick up their order themselves will receive a 10% discount.

Payment is currently made in cash upon delivery. It should now also be possible to pay for online orders with a valid credit card.

Customers

The customers of the delivery service are currently managed in a file box, which contains the addresses of all people who have been delivered so far. The index cards are sorted by phone number. When a customer calls, the address is searched for via their phone number. Customers who have not paid are also managed there. New customers are entered manually. When recording, it is checked whether the customer is in the delivery area of the delivery service, or whether the travel time to the customer makes the delivery unprofitable.

Employees

The delivery service employs a whole range of employees, including many students from the Rosenheim University of Applied Sciences. The students are usually used as drivers and deliver the orders. However, students are not available every evening, so scheduling shifts must take student availability into account. Several cooks are also employed in the delivery service. It should be noted that cooks are scheduled to work 40 hours a week. Other assistants who take orders by phone are occasionally deployed. The working hours of the cooks, assistants and drivers are currently recorded on a trust basis: each driver fills out a time recording sheet at the end of the day. Working hours are used for payroll at the end of the month.

Alessandro is currently doing his personnel planning with a large board in his office. He wants a solution that can simplify his planning.

Accounting

The delivery service has a simple accounting system that can import text files. The BuChi accounting system creates the daily, monthly, and annual cash accounts for the delivery service.

At the end of the day, a cash register compares the orders with the money received (daily turnover). The sales of the last year can be assigned to each customer to reward particularly good customers.

Catering service

In addition to the delivery service, Alessandro and Silvio want to set up a catering service to cater for smaller events, such as family celebrations. There is also an additional catalog with various suggestions for buffets. The catering orders are always designed individually by the customer. The customer coordinates the sequence of dishes and quantities with the service. The dishes are delivered to the customer by a service cook before the event.

The service has a limited capacity for creating dishes. A maximum of 50 hot dishes for buffets are possible in the kitchen.

Additional Information

Framework

Alessandro and Silvio would like to use the first functions of the software as soon as possible. The most important thing is addressing management with customer data. After that, the orders should be able to be entered electronically.

Menu and prices

Pizza Margherita	Tomato sauce, freshly grated Edam	€4.40
Pizza Funghi	Tomato sauce, freshly grated Edam, fresh mushrooms	€5.30
Pizza Salami	Tomato sauce, freshly grated Edam cheese, salami	€5.30
Pizza Prosciutto	Tomato sauce, freshly grated Edam cheese, shoulder of ham	€5.30
Pizza Tonno	Tomato sauce, freshly grated Edam, tuna, onions	€5.90
Pizza Hawaii	Tomato sauce, freshly grated Edam cheese, ham shoulder, pineapple	€5.90
Pizza Regina	Tomato sauce, freshly grated Edam cheese, fresh mushrooms, ham shoulder, pepperoni	€6.30
Pizza Quattro Formaggi	Tomato sauce, four different savory cheeses	€6.90
Pizza Frutti di Mare	Tomato sauce, freshly grated Edam, seafood, shrimp	€6.50
Pizza Quattro Stagione	Tomato sauce, grated Edam, mushrooms, ham shoulder, artichokes, pepperoni, olives	€6.90
Pizza America	Tomato sauce, freshly grated Edam, corn, salami, onions, pepperoni	€6.30
Pizza Gorgonzola	with fresh tomatoes, Italian gorgonzola, spinach, and freshly grated Edam gratinated	€6.90
Pizza Spinaci	Tomato sauce, freshly grated Edam cheese, fresh mushrooms, ham shoulder, spinach, egg	€6.90
Pizza Vegetariana	Tomato sauce, freshly grated Edam cheese, fresh vegetables	€6.90

Additional articles

Edam cheese, tomato slices, tomato sauce, fresh mushrooms, peppers, onions, corn, olives, capers, minced meat, shoulder of ham, salami, anchovies, tuna, seafood, broccoli, spinach, artichokes, egg, feta cheese, mozzarella, gorgonzola, pineapple, pepperoni: €1.00 each.

6 Exercise 6: Requ. Engineer.: Capture & document requirements

... and the project for the development of the pizza delivery service system for the pizzeria **AleSi** continues. Today it is about the documentation of the context and the compact, clear presentation of the required properties. Careful work is worthwhile here because both are of central importance for coordination with the customer.

In the following, use the mind map created in the last exercise!

6.1 Task 1: Context Diagram

Create a context diagram for the delivery service system!

If necessary, identify different system versions, because not everything has to be implemented in a single version (→ incremental process model).

6.2 Task 2: Document requirements

Document the requirements determined in the last exercise for the delivery service system to be built.

a) First, create a spreadsheet (**Excel** sheet) in which to specify the following important information for each request:

- Unique **identifier** (number) so that the requirement can be identified.
- (Short / catchy) **title**, (more detailed) **description** text
- **Source** of the requirement (which stakeholder?)
- **Category** (functional, quality, framework)
- **Type** (user requirement, system requirement, component requirement, ...)
- **Stability** (high / medium / low)
- **Complexity** (high / medium / low)
- **Priority** of the requirement (mandatory, optional, future)
- **Voted** (yes / no)

b) For the pros ☺ (optional): construct your Excel sheet with suitable columns in such a way that…

- if possible, you can select the permitted entries via drop-down selection[27],
- can be filtered per column,[28]
- in the case of a mandatory requirement, all entries in the line are automatically displayed **in red/bold**,[29]
- in the case of a "nice-to-have" requirement (future), all entries in the line are automatically displayed in *blue/italics*.

c) Finally, enter the requirements identified based on the mind map in your table. Try to indicate the category of a requirement (column "Category" in your table: functional, quality, framework) in each case.

---

[27] See: Data → Data Tools → Data Validation…
[28] See: Data → Filter.
[29] See: Start → Conditional Formatting.

6.3   Task 3: Writing down requirements as user stories

As a conscientious software engineer, you make sure to obtain your information from all stakeholders[30] if possible. You convinced Alessandro and Silvio to invite suitable guests to a small "User Story Workshop."

In this workshop you have learned, among other things, …

> ... that customers also want to add ingredients to pizza orders over the Internet to be able to adapt pizzas to their personal taste. No problem for Silvio that all ingredients mentioned on the menu[31] can be chosen. However, Alessandro has the quality of his pizzas in mind and therefore interjects that the same ingredient can only be added twice. In order not to experience any surprises when the order is placed, customers would like it if, in addition to the ordered pizza, the selected ingredients (marked as "double" if added twice) and the corresponding total price are also displayed. And, finally, the "electronic addition" should of course work without much delay if possible.

a)   Create a user story[32] (at least story name and description according to the text template) with a confirmation section, in which you formulate all the above-named acceptance criteria (list with "Verify that ...") to be able to derive test cases from them.

b)   In the confirmation section, create a small demo script for a concrete example using an input-output table. At the same time, depict the situation for a third addition of the same ingredient.

Input (selection, data, action)	Output	
	Price?	Hints?
Margarita		
...		

---

[30] See exercise 2 in the last exercise sheet.
[31] See section 4.2 of the appendix in the last exercise sheet.
[32] See script page 191 ff.