Inhaltsverzeichnis

1. Analysis

1.1. Requirements Engineering

Requirement Engineering is concerned with the definition of requirements to a system that will be developed. The goal is to arrive at a set of requirements that completely describe the system, including its functionality, the constraints, what it should and should not do. The requirements guide the implementation and serve as rationale for the systems properties.[1]
There are three activities in Requirements Engineering: Requirements Elicitation, Requirements Analysis, and Requirements Management. In Requirements elicitation, requirements are collected. There are many sources for requirements to be consulted/investigated. A big portion will come from the client. If for example the client wants the system to provide a user interface, one requirement will be: The system shall provide a user interface. Other sources are for example: industry best practices, available resources, security standards or ethical codes of conduct. The requirements can be quite specific, or broader. Some requirements might stand in conflict with each other.
In the next step, requirements analysis, the requirements are refined, categorised, broken down and conflicts are solved. Requirements can be categorised into two groups: functional and non-functional requirements. Functional requirements are concerned with the functions a system should have. “The System shall provide a login function” is a functional requirement. Non-functional requirements describe the qualities a system should have. This could be accuracy, performance or security. “The system shall be modifiable to include new functionality in the future” is such a requirement. In practice, the distinction between functional and non-functional requirements can be very difficult. Requirements that contain multiple requirements can be refined by breaking them down into smaller ones. [3]A requirement posed by the client could be “The system shall have a secure login mechanism”. This can be broken down into: “The system shall have a login mechanism” and “The login mechanism shall be secure”. A broad requirement can lead to more specific requirements. When multiple requirements stand in conflict, that conflict has to be resolved.[3] For example, a system should accommodate a number of users, but it should also run on resources that cannot support that number of users. To solve this problem, the maximum number of users could be reduced.
In Requirement Management, requirement change is managed. Requirements can change, sometimes the client changes them, sometimes the environment changes. During the whole lifecycle of requirements, these changes have to be included, so that the requirements always match up with the finished product. To determine whether the System fulfils its requirements, it has to be tested and verified.

1.1.1. Goal Oriented Requirements Engineering

Goal oriented requirements engineering is an approach to requirements engineering that uses goals to elicit and analyse requirements. “A goal is an objective the system under con-

sideration should achieve"[3]. Goals can be detected by asking the questions why, how and how else. These questions lead to different levels of goals. Asking for reasons (asking why), leads to the highest-level questions, "it helps to discover the objectives and rationale behind the goals"[3]. The how question leads to lower level goals of technical nature. Asking the how else question "helps to identify the alternatives to satisfy higher level goals"[3]. These questions are part of the requirement elicitation phase.
How do requirements relate to goals? A goal is more high level. By breaking it down and refining it, requirements are formed. Requirements are more concrete and specific. For example, a goal could be: "The UI should be intuitive" and a corresponding requirement could be : "The UI should be consistent across different features".

1.2. EDAP - Ethical Deliberation in Agile Processes

EDAP – Ethical Deliberation in Agile Processes, is a schema that assists teams in the process of "document[ing] the ethical and moral dimension of their development decisions clearly and concisely, thus facilitating sound and coherent deliberation that is easy to understand and readily verifiable by third parties."
To illustrate the process, an example will be used. Microsoft has introduced a productivity score tool. It allows employers to inspect an employee's activities. The employer can see how much an individual uses tools like email or how they contribute to shared documents. Microsoft has also filed a patent for a "meeting insight computing system". It uses data like body language and facial expression of the participants, time of day and the number of participants, to predict how high-quality a meeting will be. In addition to that, it calculates how much work the participants are doing for the meeting, in comparison to other activities, like checking their email. Both the productivity tool as well as the patent have been met with a lot of concern. Not only are the implications for the employees' privacy grave, but the proposed tools could also impact the dynamic of a workplace in many negative ways. Employee's autonomy and the trust in their ability to do what's best would be undermined and friendships between employees could be impacted by competitive behaviour. As can be seen here, there are not only consequences and moral aspects that are intuitive to recognise, but also ones that are more subtle and less easy to predict. That does not mean they are less important or serious. Therefore, in order to understand and comply with the moral and ethical aspects, developers need a process that provides support and structure to their deliberation. EDAP fulfils this requirement. How would it handle a proposed tool like the productivity tool mentioned above?
The first step is to collect all aspects that have to be considered for the system. This includes the description of the universe that the system will be placed in and the general ethical values that have to be considered. The description of the universe is important to understand what impact the system will have on the structures it is placed in. In the Microsoft scenario, the impact on the social aspects of a workplace could be understood better and taken into account. In the next step, the stakeholders are listed. For each stakeholder, the values represented by that stakeholder are specified. This does not only mean asking stakeholders what their

interests are, but analysing what their values would be in general. This can be done with the help of research that has already been done in this area. This way, the stakeholder analysis is more complete and consistent. In our example, the stakeholders include the managers that use the tool, the employees whose data will be analysed and the company. The company and managers' interest is to increase productivity. The employee might be interested in being more productive, but also wants to have their privacy protected.
Finally, the technical aspects of the system are inspected. What are the technical biases? What are the possible technical strategies and what consequences could emerge? The technical strategy that microsoft has chosen is to monitor the employees productivity and show the information to managers or team leaders. What could be consequences that would emerge with the tool's use? Managers could start to punish employees based on their scores. They could also start to dictate exactly how employees have to work. As mentioned above, the workplace social relationships could be impacted as well.
This first phase provides the basis for the argumentation to come. All information that is available and relevant to the decisions to be made is collected. The division into universe, stakeholders and technical aspects encourages the developers to consider all aspects of the system and universe. This is important because the ethical deliberation requires the analysis of the impact of the system on its environment, as opposed to viewing the system as isolated from the environment and ethical values. A developer might implement a tool like microsoft's meeting analysis and only think about the way inefficient meetings hinder the workflow and block time. This is obviously not enough, because the analysis of employees body language or software use is a serious problem for privacy. EDAP helps to inspect all aspects of the system and thus avoid such situations.
In the next phase, conflicts between values, goals and interests are explored. For each conflict, the possible options for action are presented. From the information collected before, the developers can deduce which values lie in conflict with each other. In the productivity tool, conflicting interests are for example the right to privacy against the increase of productivity. Here it is important to not only name the conflicts but to also categorise them. In this example, it would be the individual value of privacy against the social value of productivity.
Each conflict is resolved in the next phase. For the options of action, pros and cons are collected. From this information, it is decided whether to stop and choose a different solution to analyse or to go forward. In that case, the technical feasibility is analysed. The final step is testing if the finished System holds the standards that were defined.

1.3. EDAP in the development process

EDAP in Requirements Engineering

For including EDAP in the development process, it can be helpful to inspect which part of the development process it is most similar to or which part it can be integrated into. The most promising candidate is Requirement Engineering, because, like EDAP, it is concerned with the requirements of the system and their analysis.
To analyse where EDAP stands in relation to RE, it is sensible to first look at what RE does.

Requirements Engineering defines Requirements by collecting and analysing them, to arrive at a set that completely describes the system without contradictions. Requirements describe the system and define what it should and should not do. Therefore, requirements are normative: “Normative ethics [] involves substantive proposals concerning how to act, how to live, or what kind of person to be.”[hK1]. This means that the RE process is an ethical practice. Ethical does not necessarily mean that a requirement has a moral motivation, requirements can stem from goals that have no moral aspect, too. The RE process in practice generally focuses mostly on requirements with no moral component, such as functional requirements.
EDAP is also focused on norms, but in contrast to RE in practice, it focuses on morally motivated norms. These norms are most similar to goals, as they are described in goal oriented RE, especially in regards to conflicts between moral norms: “Goals have been recognized to provide the roots for detecting conflicts among requirements and for resolving them eventually.”[3] Moral norms are high level and universal. We now know that RE and EDAP both process norms, now the question is: is the EDAP process similar to that of RE?
RE starts with the requirements elicitation phase. Here, all requirements are collected. EDAP has a similar first phase: all aspects, interests and values of the universe, the system and the stakeholders are listed. This means that both EDAP and RE have a first phase which has the goal of collecting all information relevant to the system. The next phase in RE, requirements analysis, transforms the requirements by refining them and solving conflicts between requirements. EDAP also resolves conflicts in its next phase, but it does not focus on refining the other requirements. The last phase of RE is the requirement management phase. It includes verifying the requirements and monitoring and integrating changes to the requirements that occur during the development process. This is necessary to guarantee that the system will fulfil all its requirements. In EDAP, the last step is to verify and test whether the decision made in the conflict resolution is fulfilled. EDAP and RE do similar things in this phase, too.
As discussed before, the moral ethical aspects of a system need to be fulfilled in order to guarantee an ethically sound system. RE should process all norms, EDAP processes all norms of moral nature, a subgroup of norms. It would therefore make sense to integrate EDAP as a subprocess of RE.

EDAP in agile methods

EDAP now needs to be integrated into the development process. Agile methods have become established as a good way to do stuff. Since EDAP will be established as a part of RE, the first step is to analyse how Agile methods handle requirements. As a representative of agile methods, scrum will be examined. In Scrum, requirements are collected in the product backlog. Each iteration, a set of items from the product backlog is selected to be worked on. In the sprint planning meeting, requirements are refined. It does not make sense to analyse all requirements before that, because agile methods specialise on constant change, including requirements change.[2] Requirements are therefore only analysed before their respective sprint. The first part of EDAP should be worked on in the sprint planning meeting. Here, specific features, goals and requirements for the sprint are available and can be evaluated. The phases one to seven of EDAP, up to the technical feasibility, should be processed. This

leaves the last phase: Verification and validation, which is well suited to the sprint review meetings.

2. Agile tools

2.1. Criteria for Agile Tools

To evaluate the agile tools in regards to working with ethical requirements, it is necessary to define some criteria. In [1], RE activities are mapped onto Scrum activities. This can be translated into a set of activities an agile tool should support in order to qualify for the RE process. In addition to these activities, the deliberation requires traceability. “Requirements traceability refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction”[1]. The deliberation should be consistent.
From the table we can gather that the requirements are stored in the Product Backlog. This means that a tool must have some form of a backlog to store the requirements. If a backlog is available, the way requirements can be modelled should be inspected. Is there an option to mark a backlog item as a requirement or can the backlog only contain tasks? In addition to that, the usability of the backlog in regards to requirements elicitation should be inspected. Are there helpful functions to assist the brainstorming of requirements or goals?
The requirements analysis includes prioritizing and refining the requirements and the resolution of conflicts. When refining a goal or requirement, the new requirements should be linked to their parent. This ensures that a requirement is traceable to its origin. The tool should therefore support the prioritisation and refinement of requirements and linking requirements. Solving a conflict requires the documentation of the decision made in the process and the argumentation that lead to it.
In the review meetings, the requirements are validated. This means that the system should fulfil a requirement. If the requirement is satisfied or not should be documented, which means that the backlog needs to support marking a requirement as satisfied or not.
During the sprint and the planning, requirements are added or deleted. Another part of requirements management is monitoring the requirements. This means that the backlog should support these actions. Resulting Criteria:

the tool provides an implementation of the product backlog or something similar the backlog provides functionalities for:

- adding and deleting items
- editing items
- prioritising items
- differentiating between goals/ requirements and tasks
- linking items to other items
- monitoring progress of items

- marking items as fulfilled or unfulfilled

the tool provides functionalities for documenting the conflict resolution for requirements If a tool fulfils these criteria, it only means that RE could work on it, it doesn't guarantee an easy and smooth workflow. That is why, in addition to these criteria, the tool should provide functionalities that make the RE process easier and more intuitive, for example the option to visualise links between requirements or assisting questions that help in the requirement elicitation. The criteria defined above concerns RE in agile processes. For EDAP in agile processes, there are some additional requirements to the tool.
As discussed before, EDAP focuses on the conflict resolution. This means that the criteria for the agile tools need to consider the conflict resolution phase. The decisions and its reasoning need to be documented in the tool. The EDAP process will be followed by developers, not ethicists. They might not have significant experience in analysing ethical aspects of a project. It would therefore be beneficial if the tool offers helpful functionality to follow the process. Generally, the documentation should be more in-depth and more structured than in normal RE. To the criteria above, the following criteria are added:

- the tool offers assistance and structure for the EDAP process
- the tool offers functionality to thoroughly document the deliberation

2.2. Evaluation of Agile Tools

To find a selection of agile tools to test, seven articles about the best agile/scrum tools were collected and the most often noted tools were selected. This provided a list of 5 tools:

- Jira
- Pivotal Tracker
- Vivify Scrum
- Wrike
- Monday.com

The tools were examined in regards to the criteria defined before. The results for the criteria that are necessary for RE are depicted in the table below.

Jira

Jira centers around issues. An issue can have the attributes project, issue type (task, story, bug, epic), summary, description, assignee, labels, sprint, story point estimate, reporter, attachment, linked issues, flagged. issues can be organised into sprints and a backlog. On a board, the status can be viewed. Documentation can be done via Confluence pages, which can be customised, too. There is a Page template for product requirements.

tool name	add and delete	edit	prioritise	link tasks	status	req. vs task
Jira	xx	xx	x	xx	xx	x
Pivotal Tracker	xx	xx	xx	xx	xx	x
Vivify Scrum	xx	xx	xx	xx	xx	
Wrike	xx	xx	xx		xx	
Monday.com	xx	xx	xx	xx	xx	

Tabelle 1 xx - fully satisfied, x - partly satisfied

Pivotal Tracker

Pivotal Tracker centers around stories. They can be added to epics. Attributes are: Story Type, points, requester, owners, blockers/impediments, labels, code, comments and tasks. Stories can be added to the current iteration. Integrations like gitlab or Jira can be added.

Vivify Scrum

Vivify Scrum organises items into a backlog and sprints. Items can have the attributes: estimation, comments, files, flags, subitems, checklists, events and labels. They can be exported into json or csv files. Stats show the statistics for the board. Each board also has a folder for documents related to it.

Wrike

Wrike centers around tasks. They can be organised into phases, which makes it suited to traditional projects, too. Tasks have the attributes: Date, linked files, assignee, status and importance. Tasks can be organised into custom spaces, projects and folders. They can be viewed in a list, board, or table.

Monday.com

Monday.com centers around “elements”. They can have the attributes: title, subelements, assignee, status, priority, estimation and epic. Other attributes can be added, such as linked documents. Team members can comment on elements. Elements can be organised into sprints and a backlog or in a project, in custom groups. There are different views to choose from, such as Sprint planning or as a Gantt Diagram.
Features for documentation come only in the form of linking documents from the cloud or your computer and exporting a board as an excel file. Integrations of other software like gitlab or microsoft teams can be added.

Conclusion Agile Tools

In General, the tools focus on the organisation of tasks and issues. Functionality like adding, deleting, editing, prioritising and marking status are available. From a management point of view, the tools provide good functionality. From a RE perspective, there could be more. Although the backlog and its items are well implemented, the differentiation between tasks and requirements can only be made in pivotal tracker and jira, but requirements can only described as stories. Traceability can be realised by linking tasks to each other, which can be done in all tools but Wrike. Missing from all but Jira is the ability to document the RE process. By linking documents or integrating other software, documentation is still possible, but it is less easy than in Jira. Jira provides Confluence Pages, where custom or premade templates can be used for documentation. This makes Jira the most suitable for RE.

The lack of options for documentation are a hindrance for EDAP, too. Here, the best tool would be Jira, because the premade templates include one for the documenation of decisions. It would be possible to document EDAP, but a custom template would be better suited. There are no features that can make the EDAP process easier for developers to follow.

In conclusion, RE and EDAP can be executed on some of the tools like Jira, but it would require adapting to the features that are offered. This is not efficient and creates friction that would demotivate developers from following through with EDAP. A solution would be to extend a tool with more functionality to make the process easier and smoother. Jira did not only prove to be the best suited out of the tools, it also offers the option to add apps to its tool. These can be developed using a framework provided by atlassian.

3. Kapitelüberschrift

3.1. Unterkapitelüberschrift

3.1.1. Absatzüberschrift

Dies ist die Vorlage für eine wissenschaftliche Arbeit nach dem Corporate Design der Technischen Universität München (TUM). Die Vorlage ist für „MiKTex 2.9“ kompatibel.

Bitte geben Sie Ihren individuellen Text an den vorgesehenen Stellen ein und beachten Sie die Formatvorgaben des jeweiligen Lehrstuhls oder der Prüfenden zum inhaltlichen und formalen Aufbau der wissenschaftlichen Arbeit. Achten Sie grundsätzlich auf ein angenehmes Erscheinungsbild für den Leser und dass ein 1,5-facher Zeilenabstand und am Rand genügend Platz für Korrekturen eingehalten wird[1,2,3].

Grundsätzlich sind die Schriftarten Arial und Times New Roman, sowie die Neue Helvetica zulässig. Der Text ist links ausgerichtet und in Blocksatz gesetzt. Auszeichnungen der Schrift können durch Fettung, Schrägstellung und Unterstreichung erfolgen. Farbige Schrift sollte nur in Ausnahmefällen oder Grafiken zum Einsatz kommen.

Abbildung 1 Titel, Autor

[1] Bitte beachten Sie die Zitationsvorgaben Ihres Prüfers.
[2] Bitte beachten Sie die Zitationsvorgaben Ihres Prüfers.
[3] Bitte beachten Sie die Zitationsvorgaben Ihres Prüfers.

Passen Sie gegebenenfalls die Ränder an die Vorgaben Ihres Prüfers an und beachten Sie dabei, dass das Logo der TUM sich oben rechts innerhalb der Ränder, auf der Titelseite befindet. Für die Titelseiten stehen separate Vorlagen zur Verfügung.

Zur Definition von Abk. (Abkürzungen) erstellen Sie für die gewünschte Abkürzung einen Eintrag in der Datei `Abkuerzungen.tex` und referenzieren sie ihn mittels `\gls`; diese tauchen nach einem Lauf mit `latexmk` im Abkürzungsverzeichnis auf. Beispiel:
Definition in `Abkuerzungen.tex`**:** `\newacronym{abk}{Abk.}{Abkürzungen}`
Referenzierung: `\gls{abk}`

Für weitere Informationen zu Glossaren und Abkürzungen siehe die Dokumentation des Pakets `glossaries` und die entsprechenden Abschnitte in den Vorlagendateien.

3.1.2. Aufzählungen

- Dies ist die Standardaufzählung
  - Dies ist die nächste Ebene der Aufzählung

3.1.3. Nummerierungen

1. Erster Punkt der Nummerierungen
   a. Unterpunkt der Nummerierungen

Abbildungsverzeichnis

Tabellenverzeichnis

Tabellenvarianten

Überschrift Tabelle 1

Spalte 1	**Spalte 2**
Nummer 1	Nummer 2
Nummer 1	Nummer 2
Nummer 1	Nummer 2

Tabelle 2 Beschreibung

Überschrift Tabelle 2

Spalte 1	**Spalte 2**
Nummer 1	Nummer 2
Nummer 1	Nummer 2
Nummer 1	Nummer 2

Tabelle 3

Überschrift Tabelle 3

Spalte 1	**Spalte 2**
Nummer 1	Nummer 2
Nummer 1	Nummer 2
Nummer 1	Nummer 2

Tabelle 4

Tabellenvarianten 2

Überschrift Tabelle 1

Spalte 1	Spalte 2
Nummer 1, mehrzeilig in Schriftgröße 9 pt	Nummer 2
Nummer 1	Nummer 2
Nummer 1	Nummer 2

Tabelle 5

Überschrift Tabelle 2

Spalte 1	Spalte 2
Nummer 1	Nummer 2
Nummer 1	Nummer 2
Nummer 1	Nummer 2

Tabelle 6

Überschrift Tabelle 3

Spalte 1	Spalte 2
Nummer 1	Nummer 2
Nummer 1	Nummer 2
Nummer 1	Nummer 2

Tabelle 7

Literaturverzeichnis

[1] Andrea De Lucia and Abdallah Qusef. Requirements engineering in agile software development. *Journal of emerging technologies in web intelligence*, 2(3):212–220, 2010.

[2] Alberto Sillitti and Giancarlo Succi. Requirements engineering for agile methods. In *Engineering and Managing Software Requirements*, pages 309–326. Springer, 2005.

[3] Axel Van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Proceedings fifth ieee international symposium on requirements engineering*, pages 249–262. IEEE, 2001.