DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Digitizing EDAP - A Tool for Ethical Deliberation in Agile Processes

Hannah Kühn

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Digitizing EDAP - A Tool for Ethical Deliberation in Agile Processes

Digitalisierung des EDAP - Ein Tool für Ethische Deliberation in Agilen Prozessen

Author:	Hannah Kühn
Supervisor:	Prof. Dr. Alexander Pretschner
Advisor:	Severin Kacianka, Niina Zuber, Jan Gogoll
Submission Date:	15.03.2021

I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 15.03.2021 Hannah Kühn

Abstract

Using ethical deliberation in software development has become very important, as there have been moral issues with many software systems. This work integrates EDAP - ethical deliberation in agile processes into agile software development processes. The relationship between requirements engineering and ethics is explored and the place where ethical deliberation should be done in scrum, an agile framework, defined. A set of tools for agile software development is rated by their ability to support EDAP with the result that the best-suited tool is Jira. Jira is an agile development tool that can be used to plan and track software development. To make Jira optimal for EDAP, a template is constructed. It can be used to follow EDAP directly in Jira. In addition to that, an app that is designed to be used in Jira has been developed. It provides a checklist where users can select statements that apply to the current system. The app then presents information on possible ethical aspects and issues that are relevant to the project. This information can then be used to fill out the EDAP template. The implementation was evaluated with a user study where testers used the template and app and after that, filled out a questionnaire.

Contents

Abstract **iii**

1. Introduction **1**
1.1. Motivation 1
1.2. Solution 1
1.3. Contribution 2
1.4. Related Work 2

2. Place of EDAP in Software Development **4**
2.1. Requirements Engineering 4
2.2. EDAP - Ethical Deliberation in Agile Processes 5
2.3. EDAP in the development process 7
2.3.1. EDAP in Requirements Engineering 7
2.3.2. EDAP in agile methods 8

3. Agile tools **10**
3.1. Criteria for Agile Tools 10
3.2. Evaluation of Agile Tools 12

4. Extending Jira to support the EDAP Process **15**
4.1. Proposed Solutions 15
4.1.1. EDAP Template in Confluence Pages 15
4.1.2. Checklist App 15
4.2. Implementation of Checklist App 17
4.3. Integration of the app and EDAP schema in the Development Process . 18

5. Evaluation **21**
5.1. Method 21
5.2. Results 23
5.2.1. Evaluation of text questions - checklist app 23
5.2.2. Evaluation of text questions - EDAP template 24
5.2.3. Evaluation of rating questions 25
5.2.4. Limitations 29

6. Conclusion **31**

A. Appendix **33**
A.1. Questionnaire answers 33

List of Figures **42**

List of Tables **43**

Bibliography **44**

1. Introduction

1.1. Motivation

Over the past years, software systems have been introduced into different areas of life, the economy and society. These systems have a great impact. Social media has changed the way we communicate and decisions are made using artificial intelligence. Often, the ethical and moral implications of software systems are only an afterthought. From YouTube's recommended feature that guides users towards more and more extreme content [19] to Instagram filters that impact adolescent girls' body image in a negative way [14], many software systems have negative moral implications. Software is developed with goals like usability, revenue or security in mind. It is only when the system is deployed and problems start to appear that ethics is taken into consideration. Applying goals like ethical correctness at the end of development is a requirement error. "The inevitable outcome of requirements errors is time consuming and costly rework, Analyst report that rework can consume 30 to 40 percent of the total effort expended on a software development project."[21] Still, ethics is often avoided until public pressure or urgent issues make it impossible to postpone any further. Trying to fix ethical and moral issues after a system has been deployed can be very hard, if not impossible. Especially if the underlying principle or goal of a system itself is problematic, the system may need to change fundamentally. There is also sometimes a negative attitude towards ethics in software development, where people are against the introduction of what they call ideology into engineering [13]. To avoid these issues, ethical deliberation should be included in the development process.

1.2. Solution

EDAP - ethical deliberation in agile processes was developed to structure, guide and document ethical deliberation. It assists its users in considering different perspectives, aspects and biases in their decision-making. While developers can use the EDAP schema by itself, it would be better if EDAP could be integrated into tools that developers use already. Bringing ethical deliberation as close to the development as possible should be the goal. There are many tools designed for aiding agile development. They help to keep requirements and tasks organised. The deliberation could be carried out using one of these tools, if possible. By using EDAP for ethical and moral decisions, the decisions can be moved away from using gut feeling or disregarding moral values altogether, towards making them in a systematic, thorough and verifiable way. Software systems will be in line with ethics from the start, instead of being molded to fit ethics after they have been deployed already. We expect that this will not only make software systems better, but also save time, effort and resources.

1.3. Contribution

First, we establish the connection between ethics and moral values and RE. What exactly are requirements in the context of ethics and how can ethical deliberation fit into the requirements engineering process? Then the integration of the ethical deliberation into the agile development process will be determined. For this purpose, the integration of RE into agile development is examined. Using this information, EDAP can be established as a part of RE in agile development.
Next, we integrate EDAP into an agile development tool. A set of agile tools is examined if they are suitable to support EDAP. A set of criteria is defined for that purpose. The most suitable tool is selected and an extension for EDAP developed. Firstly, EDAP is transformed into a template that can be filled out. The second extension that is implemented is an app that provides a checklist that contains statements about software systems. Developers can check the statements that are true for the system they are working on. The app then collects information that applies to the ethical aspects of the system and presents it to the user.
The extension is evaluated by letting testers use the extension to make an ethical decision on a fictional scenario. Their experience is polled using a questionnaire. The results are discussed and possible improvements are proposed.

1.4. Related Work

In this work, EDAP is integrated into the development process. EDAP was introduced by [27]. It is a schema that assists teams in the process of documenting the ethical and moral dimension of their development decisions, in a way that is easy to understand and verifiable. The schema is be part of this project.
Requirements engineering is important to EDAP because EDAP can be established as a part of RE. [25] explore the use of Requirements Engineering in practice with the help of surveys. The survey was first piloted in Germany and then conducted in 10 different countries with the goal of answering the questions: How are requirements

elicited and documented?, How are requirements changed and aligned with tests?, Why and how are RE standards applied and tailored? and How is RE improved?. The use of RE in practice is interesting to this project, since EDAP will be integrated into the development process in practice.

Goal oriented requirements engineering is relevant to EDAP because, as will be discussed later, goals can be related to moral values and norms, which are central to EDAP. The modelling and processing of goals can be helpful to the modelling and processing of moral values and norms. [24] reviews research efforts on the topic of Goal oriented Requirements Engineering. It is also defined what goals are, how they can be modelled and processed.

[8] examines the problems connected to requirements engineering in agile software development and proposes solutions. The problem of how requirements can be captured and specified in agile development and the lack of traceability is addressed. In the end, guidelines are established. In this work, EDAP is integrated into agile processes. The problems that occur for RE in agile methods apply here, too. The results of [8] can therefore be adopted.

In [11] the term software failure is expanded to include social, professional and ethical failure. It focuses on how incomplete Stakeholder Analysis can lead to such failure and how to improve it. The prevention of software failure in terms of ethical failure is central to EDAP. The expanded term of software failure fits very well with EDAP. In contrast to [11], this work not only improves the stakeholder analysis but the whole process of handling ethical and moral requirements.

EDAP approaches the ethical and moral decisions made in development with a schema that documents the ethical reasoning. A different approach is the fast and frugal reasoning described in [10]. Fast and frugal inference algorithms are proposed. They are based on bounded rationality, meaning they don't use or need all information to make a decision. Decisions are made using cues, that indicate what decision is best. In [9] decision-making based on heuristics is examined. It is argued, that decisions-making based on heuristics rather than rational models is well suited to situations where not all information and computational power is available. Here, fast and frugal trees are described, that can be used to make decisions. EDAP is based on rational decision-making, using fast and frugal algorithms to substitute EDAP would not be suitable because EDAP needs to process very broad topics. We will however integrate fast and frugal methods to make parts of the EDAP process more efficient.

2. Place of EDAP in Software Development

2.1. 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 [8].
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 and 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 for example. "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 [24]. 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 [24]. 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 life-cycle of requirements, these changes have to be included, so that the requirements always match up with the finished product and, the other way around, the finished product matches up with the requirements. To determine whether the system fulfils its requirements, it has to be tested and verified.

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 consideration should achieve" [24]. 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" [24]. 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" [24]. 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 users accounts should be safe" and a corresponding requirement could be : "The password should be encrypted".

2.2. 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." [27]. 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 [12]. 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 [5]. 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 and makes subtle and emerging ethical aspects visible. EDAP has been created to fulfils this requirement. To show how EDAP works in practice, we use this example.

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. "Stakeholders are individuals or groups who may be directly or indirectly affected by the project and thus have a stake in the development activities."[11]. The stakeholder analysis should not focus solely on stakeholders that are directly involved in the project. "Recent research has confirmed that inadequate identification of project stakeholders and how they are affected by a project is a significant contributor to the project's failure."[11]. 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, by collecting all information that is available and relevant to the decisions to be made. 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 privacy issue. EDAP helps to inspect all aspects of the system and thus avoid software failure as defined by [11].
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.

2.3. EDAP in the development process

2.3.1. 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. 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: Most of the time, they are broad and high level. They are also similar 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."[24].
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 sub-process of RE.

2.3.2. EDAP in agile methods

EDAP now needs to be integrated into the development process. The question is now, which development style EDAP will be integrated into. Although, as described in [8], conducting RE in agile software development brings some challenges, agile methods have become very popular and are therefore a good choice as the development style. 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. The items in the backlog are worked on in iterations called sprints. The sprints are planned in the sprint planning meeting, where the requirements that will be worked on during the oncoming sprint are selected and then refined [3]. It does not make sense to analyse all requirements before that, because agile methods specialise on constant change, including requirements change [20]. 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: "The participants assess the product increment and make the decision about the following activities. The review meeting may bring out new backlog items and even change the direction of the system being built."[3].

3. Agile tools

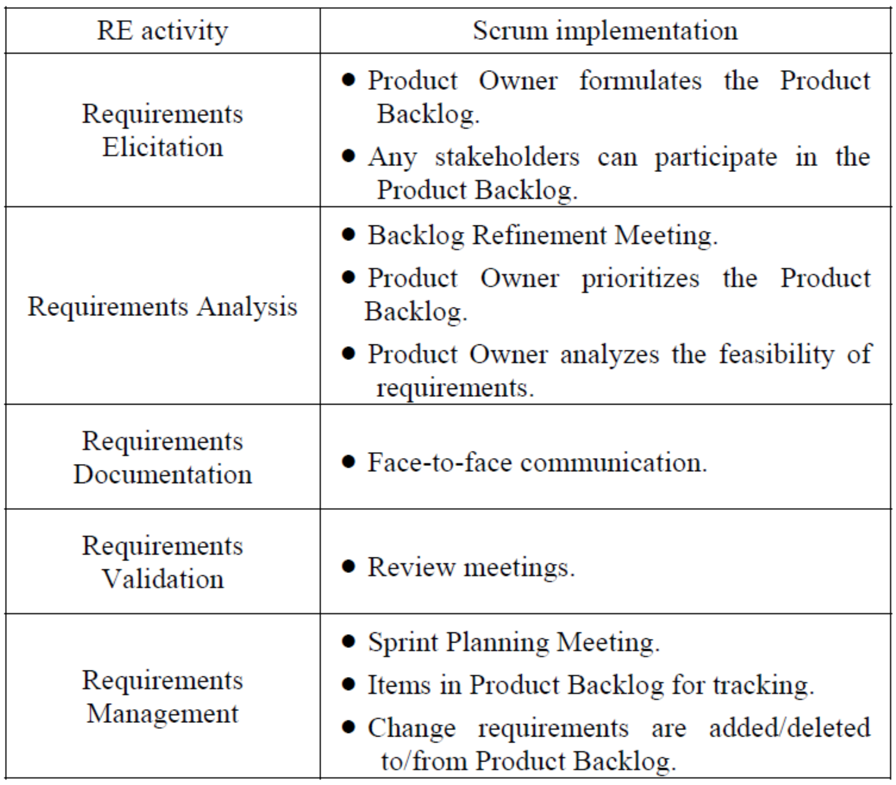
3.1. Criteria for Agile Tools

To evaluate the agile tools in regards to working with ethical requirements, it is necessary to define suitable criteria. In Requirements engineering in agile software development [8], 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"[8]. 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 tool 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. The resulting criteria are:

Resulting Criteria for RE

- *the tool provides an implementation of the product backlog*



RE activity	Scrum implementation
Requirements Elicitation	• Product Owner formulates the Product Backlog. • Any stakeholders can participate in the Product Backlog.
Requirements Analysis	• Backlog Refinement Meeting. • Product Owner prioritizes the Product Backlog. • Product Owner analyzes the feasibility of requirements.
Requirements Documentation	• Face-to-face communication.
Requirements Validation	• Review meetings.
Requirements Management	• Sprint Planning Meeting. • Items in Product Backlog for tracking. • Change requirements are added/deleted to/from Product Backlog.

Figure 1.: RE activities in Scrum, [8]

- *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 does not 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 functionality to thoroughly document and structure the deliberation*
- *the tool offers assistance for the EDAP process*

3.2. 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 [2][4][26][22][23][1][17]. 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

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	

Table 1.: Ratings for agile tools, xx - fully satisfied, x - partly satisfied

Confluence pages, which can be customised, too. There is a Page template for product requirements and decisions.

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, sub-items, 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, sub-elements, 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. For RE, there should be more functionality. 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, there. 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 directly 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 is a hindrance for EDAP, too. Here, the best tool is Jira, because the premade templates include one for the documentation of decisions. It would be possible to use this 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 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 it. These can be developed using a framework provided by Atlassian.

4. Extending Jira to support the EDAP Process

4.1. Proposed Solutions

The two requirements that the agile tools did not fulfil are:

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

4.1.1. EDAP Template in Confluence Pages

The first requirement can be satisfied by using Jira's function to build a custom template for EDAP. The EDAP schema can be directly implemented as a template, because it is already in the form of a document with tables that can be filled in. Developers can fill out this template and thus follow EDAP and document their deliberation. To make it easier to know what has to be filled into the different sections, information fields and placeholder text can be added. In the EDAP template, Jira issues and other pages can be linked. Linking the Jira issues that correspond to the feature being processed in EDAP connects the deliberation directly to the development. If developers make use of the connection between the EDAP pages, the issues and their sub-issues, the concrete tasks can be traced back all the way to the high level ethical decisions of EDAP. This and the embedding of the project pages in Jira makes the template a very suitable implementation of EDAP. Instead of being documented separately, EDAP is embedded into the management tool and thus integrated into the development process.

4.1.2. Checklist App

The second requirement that Jira does not yet satisfy could be implemented in form of an app that assists the EDAP process. As mentioned before, Jira offers the integration of Apps, that can be used directly in the Interface. It is possible, to develop a new app using Atlassian Connect.
What would a tool to support the EDAP process look like? One major difficulty in

the deliberation process is to grasp all the implications of a proposed feature. Some crucial issues might not be intuitive during development, but not considering them can have unwanted consequences. Developers are mostly not trained in ethics or ethical deliberation, so they might not have sufficient knowledge to make an ethically sound decision. In addition to that, the impact of a system on its environment can never be fully predicted because of its complexity.
To this problem the fast and frugal reasoning proposed by [10] can be applied. It is based on bounded rationality, which means that decisions are made by actors that have limited time, limited computational capabilities and limited knowledge. Developers in our problem fit this description, as the fast-paced development environment does not allow for lengthy deliberations and developers do not have all knowledge or deliberation skills. One solution that is proposed is the fast and frugal tree. Each node in the tree is a binary node representing a cue. These cues indicate what should be done. Figure 2 shows an example from a hospital environment. These trees can be transformed into layered checklists where each checkpoint represents one cue, that can have sub-checkpoints, meaning sub-cues and conclusions.

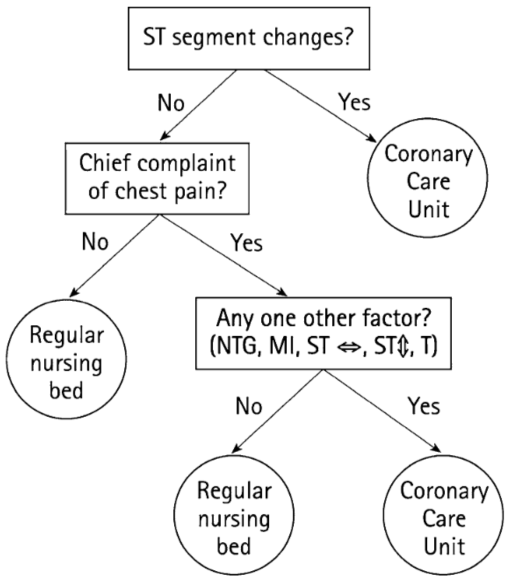


Figure 2.: Example of FF-Tree, [9]

How could such a checklist be implemented to assist the EDAP process? Trying to implement a decision tree as am ff-tree is not sensible, because it would quickly grow

too big to handle, which contradicts the goal of making decisions in limited time. Instead of trying to use the tree to make the ethical decision, it could be used to inform the developers about important information they should consider for their decision. The cues would be attributes of the project that indicate what information specifically is important. A cue could be for example: "The feature collects data from individuals". This indicates that information about the handling of personal data is relevant to the system. By ruling out cues that do not fit the project, the checklist rules out irrelevant information as well. This makes the process more efficient, because making developers read a stack of reports, guides and codes of conducts every time a feature is proposed would not work in practice.
The information that will be included in the checklist can come from many different sources. The same way that many features repeat themselves, the ethical aspects of them repeat themselves, too. For example, a productivity tool like the one proposed by Microsoft is not unique. The problems that come with it are also not unique, other productivity tools may have similar implications. Learning from past projects can help to better understand the ethical and moral aspects of the current project. The conclusion from past projects should therefore be included into the checklist. Other resources like ethical codes of conduct or guides are helpful, too. If the checklist is consequently filled and updated with information, it will form a knowledge base that can help make ethical decisions better in the future.

4.2. Implementation of Checklist App

After the developers complete the checklist, the app will collect all the corresponding information and present it. The developers can then review this information and include it in the EDAP process.

Processing information for the checklist

Adding information to the checklist information base requires processing of the data. As an example, the tool that is implemented contains information from the Data Ethics Canvas of the Open Data Institute [7]. How can the Data Ethics Canvas be converted the into a set of statements and corresponding information? From the given information, the individual risks and other things to consider have to be inspected. In what cases are they relevant? What would lead to them being relevant. For example: "could certain analysis models or automated decision-making lead to discrimination against particular groups?" [7] is only relevant if the system actually uses automated decision-making. A checkpoint statement could be: "The system uses automated decision-making" and its corresponding information for the developer: "The automated decision-making

could lead to discrimination against particular groups". In this way, other sources of information can be processed. When reviewing past projects, one can ask: Are we still alright with the decisions that were made in the development process? What went differently than expected? What caused negative outcomes? When problems are identified, the causes or conditions for that problem can be used as statements and the description of the problem as the corresponding information. When using codes of conduct, they can be processed similarly to the data ethics canvas.
The processing of data sources means extra effort every time a new source is added to the knowledge base. This is remedied by the time that is being saved by not having to go through all data every time a decision has to be made. If the knowledge base is consequently updated with new information and corrected if some parts do not prove to be useful or correct, it will grow to span many different topics.

4.3. Integration of the app and EDAP schema in the Development Process

As discussed before, EDAP is used in the sprint planning and sprint review meetings. The checklist has the purpose of giving developers indications on what they have to consider in the deliberation. The checklist will therefore be placed before the start of EDAP. In the figure below, the position of EDAP and the checklist in the scrum process are shown. In the sprint planning meeting, backlog items are selected to be processed in the next sprint. Since these items can be high level or not refined yet, requirement analysis is done to arrive at more concrete requirements. If there is potential for moral analysis, EDAP is started. The given feature is analysed in the checklist and EDAP schema. This decides, whether the feature should be implemented or not, or if a different strategy should be chosen. The figure below shows how an exemplary feature could be processed. An item is taken from the backlog and refined. One of the resulting requirements: "feature presents results to manager" raises moral objection from the developers. The EDAP process is started. First, the checklist is filled out by the participants of the meeting in the checklist app. The checklist points out risks due to the use of personal data. Then the EDAP template is used to do the deliberation. In the EDAP schema, the impact on the workplace environment is added to the potential risks. The deliberation concludes that the feature in question should not be implemented. A new solution is proposed. The issues that represent the new solution are linked in the EDAP page. This example is simplified to illustrate how the process would look in practice.

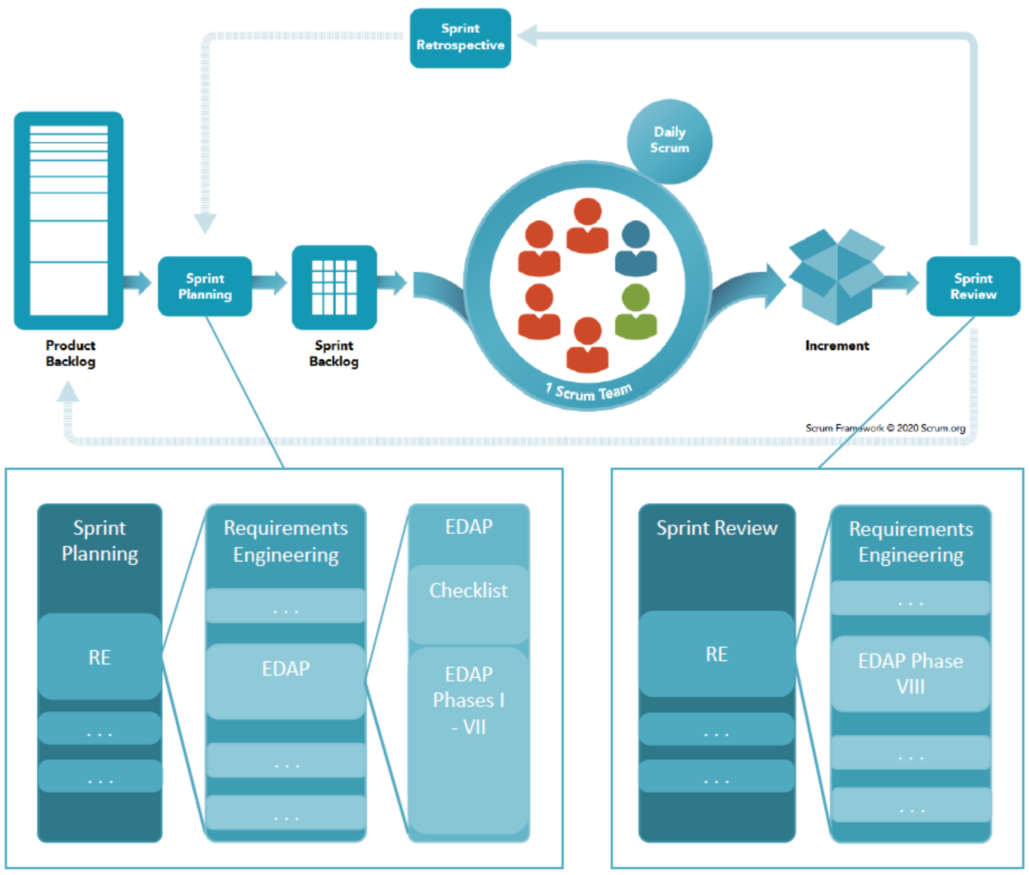


Figure 3.: EDAP process in Scrum, scrum.org with added elements

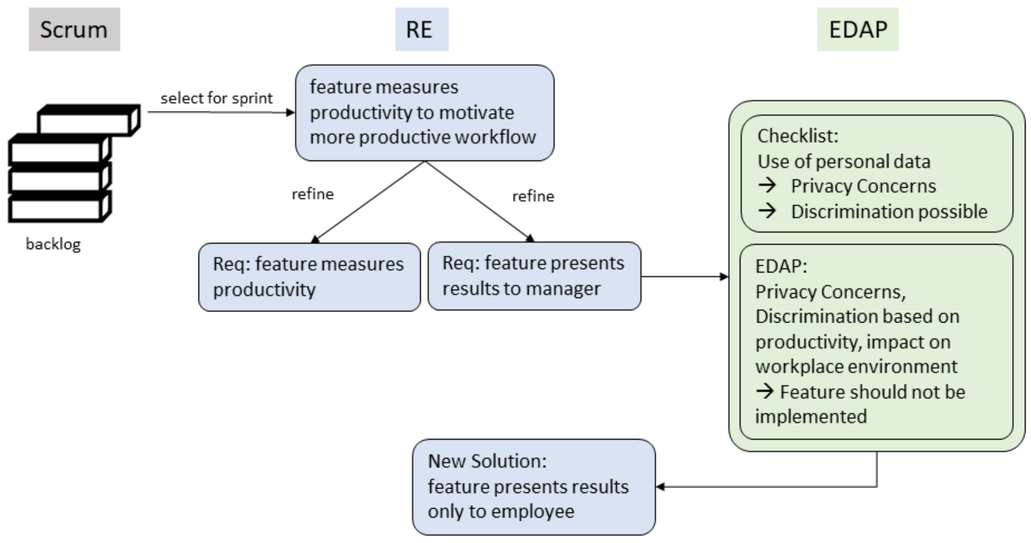

Scrum
RE
EDAP
select for sprint
backlog
feature measures productivity to motivate more productive workflow
refine
refine
Req: feature measures productivity
Req: feature presents results to manager
Checklist:
Use of personal data
→ Privacy Concerns
→ Discrimination possible
EDAP:
Privacy Concerns, Discrimination based on productivity, impact on workplace environment
→ Feature should not be implemented
New Solution:
feature presents results only to employee

Figure 4.: Simplified example of EDAP, using https://thenounproject.com/term/backlog/

5. Evaluation

To evaluate the implementation, both the checklist app and the EDAP template were tested. We wanted to find out whether the goals of the implementation have been reached. The checklist is supposed to support the EDAP process by providing helpful information in an easily usable way. [18] state, that "[e]mpirical methods are the main way of evaluating user interfaces, with user testing probably being the most commonly used method." The participants could test the usability and how helpful the checklist is to the EDAP process. The EDAP template is supposed to assist developers in documenting and making ethical decisions. It contains instructions and information fields that explain what the developers should fill in. It should be easily usable as well. These goals can be tested by developers, because they relate to the user experience. There are other goals, that cannot be tested this way. The EDAP process should enable developers to properly make ethical decisions and document them. This should lead to a reduction in Software failure from a moral perspective. If that goal is reached can only be seen when the process is used.

5.1. Method

Testing Process

The evaluation will be conducted by letting people test the implementation of the checklist app, then fill out the EDAP template and afterwards, fill out a questionnaire. As stated in [15]: "Questionnaires are a commonly used tool for the user-driven assessment of software quality and usability. They allow an efficient quantitative measurement of product features." Another possibility to test would have been to let testers use the prototype and observe them. This would have the advantage that instead of asking about the testers opinion, one could examine their behaviour. We were not able to conduct such tests because of the COVID-19 pandemic. [6] advises to prepare uniform instructions to participants. That is why participants will receive an instruction sheet that informs them of the steps they should take and provides information on the scenario they will test and the system in general.

Testing Scenario

The scenario is inspired by the productivity tool for Microsoft teams:
You are a member of a development team using Scrum. Your team is working on a work management and communication tool. Right now, you are in the sprint planning meeting. You and your colleagues have selected the following issue from the product backlog: "The productivity of employees is calculated with the data provided by the management functions of the system. The results are presented to the team manager" You and your team are not sure about the moral implications of this goal, so you start the EDAP process.

Questionnaire

After they have used the checklist and EDAP, the testers are instructed to fill out the questionnaire. We have chosen to use a mix of predefined measures and open evaluation. The predefined measures are the ease of use, helpfulness and how likely the testers are to use the tools again. These are specific aspects that can show if the desired goals have been reached. To still allow for other feedback and to gather more information, questions about advantages, disadvantages and possible improvements are added. The questions are divided into three categories: general questions, questions about the checklist app and questions about the EDAP template. The questions that can be answered on a scale have 5 categories. In general, the number of categories recommended is between five and nine [16]. Since this is an evaluation on a small scale, five categories should provide a sufficient amount of detail and more categories are not necessary.

General Questions

For the ratings on usability, it is important to know how experienced the testers are in Scrum, EDAP and Jira. For example, if the tester is not experienced in Jira, they might have a more difficult time using the template.

- Are you experienced with Scrum? (not experienced at all - very experienced)
- Are you experienced with Jira? (not experienced at all - very experienced)
- Are you experienced with EDAP? (not experienced at all - very experienced)

Checklist App

- How easy or hard was it to use the checklist app? (hard - easy)

- How would you rate the checklist in regards to filling out the EDAP template? (not helpful at all - very helpful)
- What do you think are advantages of the checklist app? (text answer)
- What do you think are disadvantages of the checklist app? (text answer)
- How likely are you to use the checklist app for future ethical development decisions? (very unlikely - very likely)
- How can the checklist app be improved? (text answer)

EDAP Template

- How easy or hard was it to use the EDAP template? (hard - easy)
- How helpful were the instructions in the template? (not helpful at all - very helpful)
- How would you rate the EDAP template in regards to making ethical decisions? (not helpful at all - very helpful)
- What do you think are advantages of the EDAP template? (text answer)
- What do you think are disadvantages of the EDAP template? (text answer)
- How likely are you to use the EDAP template for future ethical development decisions? (very unlikely - very likely)
- How can the EDAP template be improved? (text answer)

5.2. Results

The results of the questionnaire have been evaluated. The questions that were answered on a scale are plotted in bar charts. The questions with text answers were read and summarised.

5.2.1. Evaluation of text questions - checklist app

The feedback for the checklist showed, that the goals for the app have been reached. According to the testers, the advantages of the checklist are the following: Most often, the checklist was described as easy to use. This shows that the checklist format was a good choice create an accessible app. Also positively noted was that the checklist

only shows necessary information. This confirms that the reduction of the amount of information to be reviewed was implemented successfully. The checklist was described as a good starting point, that gives a good overview and indicates important points. In addition to that, it forces the user to think about data source and usage. We can see, that the checklist fulfilled its purpose of providing an easy to use entry point to the ethical deliberation with relevant information.
The feedback about the disadvantages included that there was too much text to read and that the presentation of the checklist result was not user friendly. This would only get worse when more information is added to the checklist, so it is important to improve on this issue. One proposed improvement in the feedback was to use accordion-style options instead of normal checkboxes. This would be a good solution to reduce the amount of text the user has to read at once and make the text more structured. In addition to that, the presentation of the text should be made more appealing by employing better UI design. While the amount of text seemed to be off-putting, there was also demand for more details, examples and links to more information. When adding new information to the checklist in the future, this should be kept in mind.
One concern was that the checklist is too rigid and might not apply to all systems. For the prototype, this is true. It only includes information on one topic. When the checklist is used, however, it will be expanded with more information and thus apply to more systems.
The feedback stated that information on the context and goal of the app was missing. This should be added to the app because as of now, it only contains the checklist itself. As proposed in the feedback, it should be clear what the purpose is and how the results should be used. This information could be added in a starting screen.
Another point of critique was that ethics do not bring revenue and that the app should serve developers needs and not ethical ideals. Here, it is important to consider that ethical failure of software is still software failure. In addition to that, moral values are goals, the same way for example security is. Therefore, even though ethics do not directly contribute to revenue, they are part of RE and should be of concern to developers.

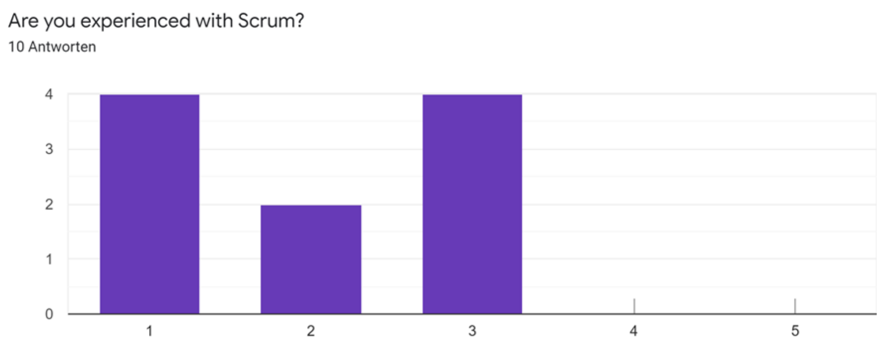
5.2.2. Evaluation of text questions - EDAP template

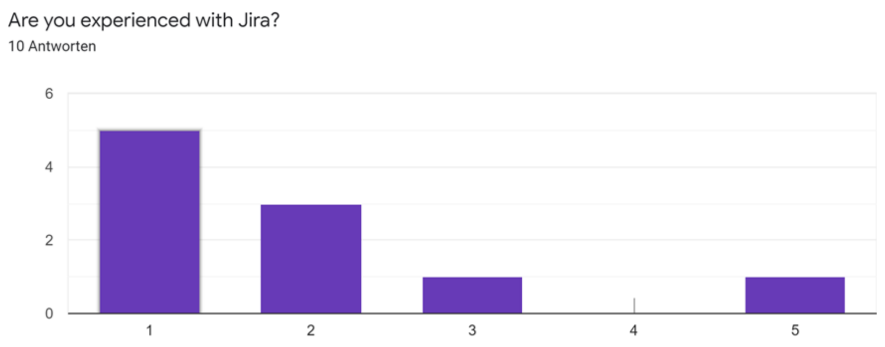
In the positive feedback it was often noted, that EDAP provides a systematic, structured and professional approach to ethical problems. Another advantage is that it raises awareness and helps not to miss aspects of the situation. In addition to that, the feedback stated that the EDAP template encouraged thinking about ethical decisions and that it is good for users without experience in ethics. Also, it helps to explicitly

consider different perspectives. The feedback showed, that EDAP succeeds in providing a structure for ethical deliberation and an opportunity to document ethical decisions. Especially the ability to encourage thinking and raising awareness has been achieved. The majority of the negative feedback concerned the usability of the template. The testers found the template difficult to use, confusing and complicated for users without experience in ethics and Jira. This is not a surprise, since most developers probably do not know terms and concepts like pretheoretical deliberation or deontological ethics. Making EDAP less complex, as suggested by some testers, would make it easier to use for beginners. The resulting problem would be that EDAP would be less suitable for complex problems. Another suggestion was to provide a sample case, where a sample problem is processed with EDAP. This is a good start, since seeing a solution in practice can help better understand how to use EDAP. In general, the consensus was that EDAP is difficult to use as a beginner. This implies that with more experience in using EDAP, it would be easier. Therefore, the goal should be to provide a good introduction to EDAP, for users that are new to it. A good way to do that would be to provide an introduction seminar or course, where a software development team learns how to use EDAP with the help of someone experienced in EDAP and ethics. Using the suggested sample case, the instructor could guide the developers through EDAP. This way, developers could learn what the used concepts and terms mean, while also seeing how it is used in practice. The instructor could provide helpful advice and feedback, so that EDAP is used optimally.
Other suggestions for making the template easier to use were to include predefined options for fields like stakeholders and using subtractive options. These would make EDAP easier to use, but they would also allow users to quickly select a few options without taking time to reflect.
One tester suggested that developers are normally not concerned with high-level problems, but instead just follow instructions by management and architects. EDAP should consider the perspective of developers more. The assumption that developers are not concerned with these problems may apply to some projects, where normal hierarchy is used. In this case, EDAP is integrated into agile processes, though. In agile software development, the team organizes itself and the decisions concerning what should be done are made by the team [3]. The traditional hierarchy, where managers decide what to do and the team follow those decisions are not applicable. Thus, developers need to be involved in EDAP.

5.2.3. Evaluation of rating questions

The testers were not experienced in scrum, Jira and EDAP. The average experience was 2, 1.9 and 1.5. This may have contributed to the difficulties in using the EDAP template.





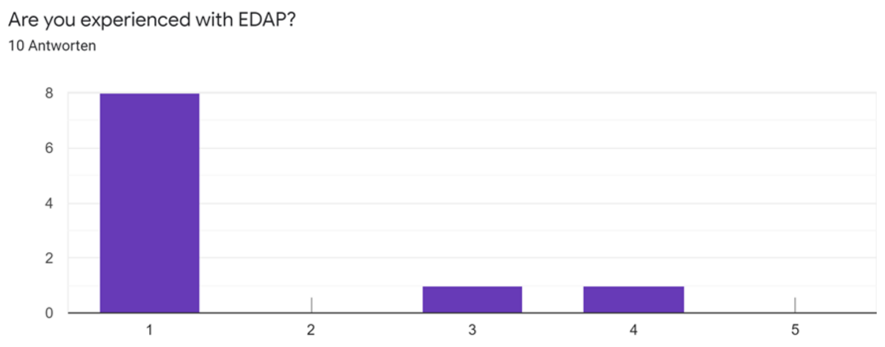
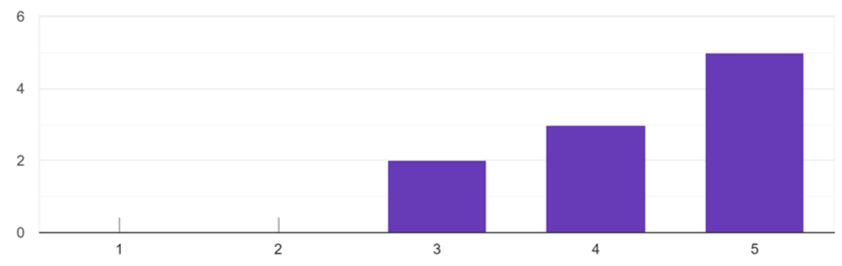
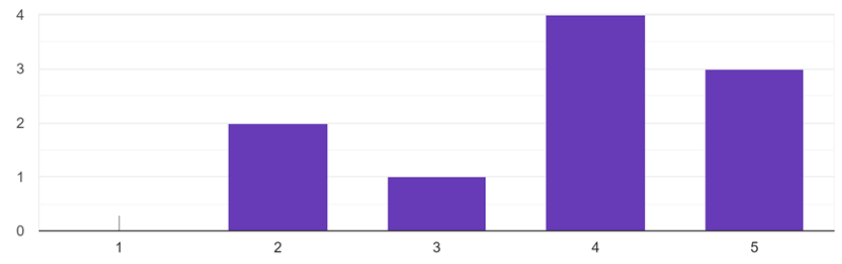


Figure 5.: Questions on general experience

How easy or hard was it to use the checklist app?
10 Antworten



How would you rate the checklist app in regards to filling out the EDAP template?
10 Antworten



How likely are you to use the checklist app for future ethical development decisions?
10 Antworten

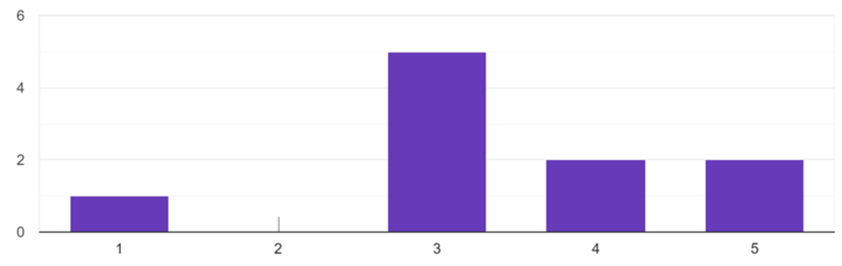


Figure 6.: Questions about the checklist app

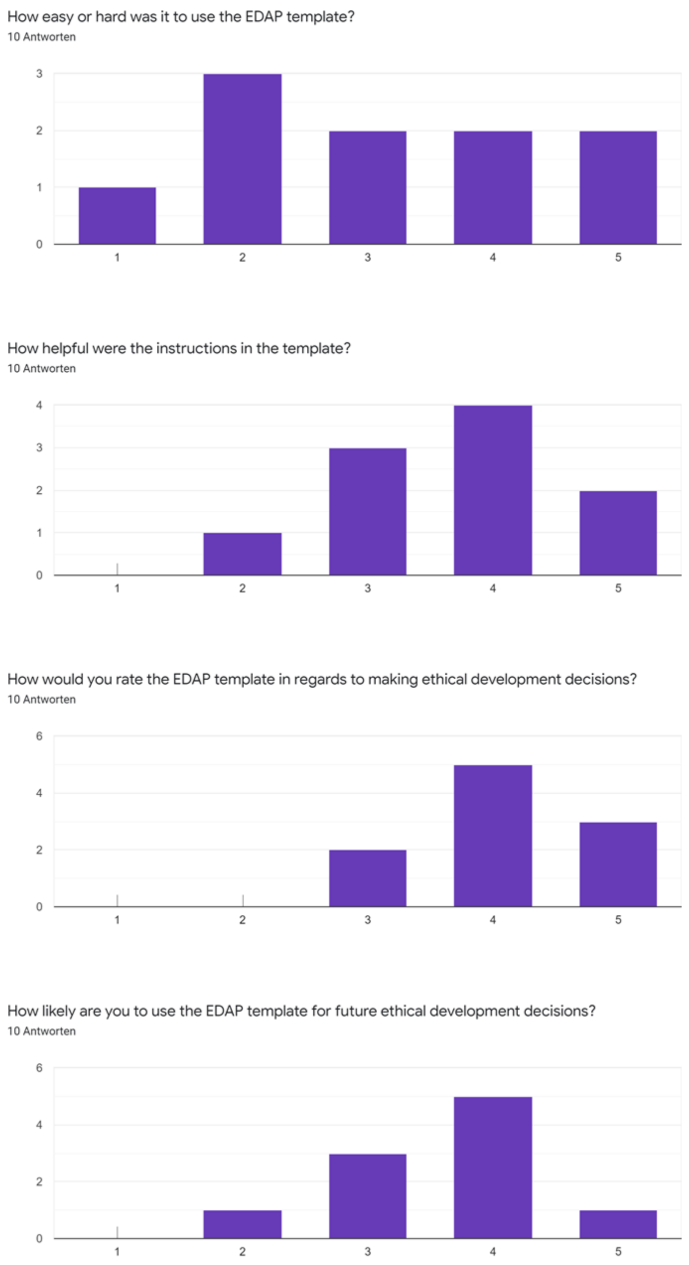


Figure 7.: Questions about EDAP

The checklist app was rated easy to use (average 4.3) and helpful for using EDAP (average 3.8). The testers were a little likely to use it in the future (average 3.4). A reason why the stated likelihood was not greater could be that the provided example was relatively simple. That may have made the information in the checklist obsolete. As one feedback noted, most abstract conflicts may already be known to the developer. Also, the checklist prototype contained only information from one topic and one source. When there is more information that is applied to more complex problems, the checklist should prove to be more useful.
The EDAP template was rated neither easy nor hard to use (average 3.1). Its instructions were rated a little helpful (average 3.7). This is surprising, given that the ratings for ease of use were not good. It seems that the instructions were helpful, but they are not the right tool to educate users on the proper use of EDAP. The mentioned introductory seminar should solve the usability issues. EDAP was rated to be helpful for ethical decisions (average 4.1) and testers were a little likely to use it in the future (average 3.6). This could be connected to the difficult use. The fact that EDAP was rated helpful for ethical decisions despite the difficulties testers had attests for its effectiveness.
In general, the testers were less likely to use both the checklist and EDAP compared to their ratings for helpfulness. A reason for this could be that an ethical deliberation means an extra step in requirements engineering that is usually not done. But, as discussed before, it is a necessary step.

5.2.4. Limitations

Due to the limited amount of people that could be asked to participate in the testing, only 10 people took part. The participants were invited by email to take part in the evaluation. They were people that study or work at the university, so it is possible that some were students who do not have much experience in software development yet. Their perspective on the app and EDAP could be less realistic than that of a working software developer. The perspective of the testers could also be biased against ethical deliberations, because instead of already using a system for ethical decisions to compare EDAP to, there is no system in place most of the time. The small sample size meant that the data could not be analysed by experience in EDAP, Jira and scrum, because the data would not have been significant enough. If there were more participants, the relationship between for example the experience in Jira and the ratings on ease of use could have been examined.
The testing only showed the opinions of the testers after a single use of the prototype. This means that the actual impact of EDAP and the app on ethical decisions could not be measured. For that, the tools would need to be implemented into real scenarios over a period of time.

The amount of options in the questions with a scale was five, so a neutral option was available. In the question "How likely are you to use the checklist app for future ethical decisions?" the neutral option was the most selected. If the scale had been reduced to 4 options, the testers would have been forced to make a decision. That may have delivered a more clear result.
The questions included two about the disadvantages and proposed improvements for the checklist and EDAP. Some improvements were entered into the disadvantages and vice versa, so the two questions could have been combined into one.
At the end of the EDAP testing, the users were informed that they can save their filled out template if they want to. The testers mostly just entered short sentences or bullet points and only half of them saved it. If the testers were instructed to thoroughly fill out the template, the resulting pages could have been examined and included in the evaluation.

6. Conclusion

We have shown that ethical deliberation can be integrated into the agile development process. First, it was established, that requirements engineering is an ethical practice: The normative nature of requirements means that all requirements are, in theory, ethical. EDAP is concerned with the moral requirements to a system. For this reason, it should be a part of the requirements engineering process. Then, the integration of EDAP in agile development was established. Using information on how RE can be implemented in agile development, we decided that the deliberation should be done in the sprint planning meetings and the verification step in the sprint review meetings.
In the next step, agile tools were evaluated on their ability to support RE and EDAP. The criteria included if the tool can be used to handle requirements, decision making and documentation of these decisions. We also explored, whether the tools provide any helpful functionality that could make ethical deliberation easier. The results showed that the agile tools are mostly focused on task management and are mostly lacking functionality to make decisions and handle requirements specifically. Options for documentation were not a part of most tools except for Jira. Jira was chosen to be the most suitable out of the tools. It has good functionality for handling requirements and the addition of Confluence pages means that documentation is easy and available. In addition to that, it is possible to develop apps that can be used directly in Jira. Jira fulfilled some, but not all requirements that were deemed necessary for a good use of ethical deliberation. Thus, we proposed the following extensions to Jira: A template (or blueprint) for EDAP that can be used for confluence pages and a checklist app to make the start of the ethical deliberation easier. The EDAP template would make it possible to do EDAP directly in Jira. The checklist app was proposed to give indications and provide important information on the system and its ethical and moral aspects. We also explained how information can be transformed to be included in the knowledge base that is the core of the checklist app. Both the template and the checklist app were implemented.
Finally the template and app were evaluated. The evaluation showed that the checklist is a helpful start for the EDAP process. It provides a good overview and indicates important points to consider. It was proposed, that to improve the checklist, more information on different types of projects and ethical and moral aspects have to be added. The display of the results should be improved to make reading them more

appealing. One option is to add drop-down information cards to each checkbox instead of showing all information after submission. In general, the UI should be designed to be appealing and engaging. The improvements that were proposed for EDAP are: The instructions in the template are not enough to guide first-time users. As a solution, an introductory seminar was proposed.
The evaluation showed, that EDAP has proven to be systematic and well structured. It enables users to make well thought out decisions and helps to consider all aspects of a system. By extending Jira to support EDAP, ethical deliberation has been made easy and accessible. It makes ethical deliberation a part of development, instead of a being detached from it. It remains to be seen if its use prevents ethically bad software in the long run.

A. Appendix

A.1. Questionnaire answers

What do you think are advantages of the Checklist app?

- Not showing all info in advance
- easy to use, not much effort needed to check for all dimensions of ethics,
- Guidelines for ethical software engineering, good starting point
- easy and direct way to get initial tips for general ethic problems
- Easy to use solution so that user is forced to think about data source and usage.
- Including only the necessary moral considerations into the design process
- good indications for better understanding
- Managers are lazy, they like to fill out checklists to feel useful.
- Easily get a good overview on important points to consider - all done by checking the relevant checkboxes.

What do you think are disadvantages of the Checklist app?

- Tedious, unclear, why not just have accordions that users can click? No value in the checklist function. Also too much text at once.
- bias introduced by the app (people use only the presented examples, other dimensions are dropped), mucht text at some point might lead to skipping the content.
- Ethics don't bring revenue and for this the results of the checklist app are a lot to read. only for getting edap process started not solved
- App itself is not really descriptive (e.g. context/goal of the app)

- Most of the abstract conflicts may already be known to the developer
- somewhat cursory, few descriptive examples
- Could have been a word-template, except for the linking issues stuff.
- rigid, might not apply to all systems

How can the Checklist app be improved?

- Start from developer needs, not from ethical ideals.
- In the explanatory texts lists including more examples would be nice.
- More human friendly user interface, less text or more beautiful representation of text
- links in solution text to further resources
- Add more description on the app context. How are the results used later?
- more details, more examples
- space to enter would be nice between sections. Sometimes i lost my cursor or added a column in the table with the keyboard shortcut. On single-line table cells, if i want to keep the grey text in, i would like to set the cursor to the end of a line (probably Jiras fault though)
- Have more points, some sub-points should only appear if the parent-checkpoint is marked

What do you think are advantages of the EDAP template?

- Raises awareness of potential problems; allows explicitly considering perspectives
- people are encouraged to think about ethical decisions on their own
- Well structured, professional approach to ethical problems in software projects
- good exercises to get thinking
- Help to discuss and organize thoughts about ethical implementations
- Systematic way of tackling the moral considerations
- very good structure, helps not to miss anything

- good for beginners that have no idea
- hvaing a fixed template to go through helps with not forgetting certain aspects

What do you think are disadvantages of the EDAP template?

- Can be intimidating if not known; not clear what is "correct" or not
- some parts are a little bit abstract, might lead to careless completion
- Maybe a bit complicated for someone that has never had anything to do with ethical decisions
- ui a bit confusing at first
- User need experience in ehtical decision making. Hard to use for "new user" without experience in this topic
- Felt lost sometimes
- partly not quite clear what is meant if you do not have much experience yet
- Its extremely free-form. Some more hints or good/common aspects would help in not missing things – go more into the subtractive (this isn't an issue) instead of having to come up with your own stuff (e.g. in values and goals)

How can the EDAP template be improved?

- Explain more from the perspective of a developer, less an architect / project lead. In my role as developer, I don't necessarily consider these high-level problems; I'm just told what to do and do it.
- predefined answers for possible stakeholders (especially stakeholders one usually does not think about)
- If the average developer is supposed to use it I would try to reduce the level of complexity and the knowledge needed about ethics by reducing the number of free text answers and increasing the number of questions where you can choose from existing answers. For example in Phase V - Ethical System Inspection there could be a catalogue of pre-existing deontological arguments, so the user doesn't fully have to understand what deontological exactly means.
- The template is fine itself. Problems stem from the inexperience with Jira and ethical decision making.

- A bit more help e.g. examples or so would be important. Sometimes felt lost and did not knew what was asked from me or what the words meant. UI-wise it was difficult to figure out where to put the answer
- more information on the individual aspects, perhaps also provide a completed sample that is based on an example case
- whats the difference between checklist app and EDAP template? i only filled out a template document.

EDAP

Phase I - Descriptive System Analysis

Describe the system and list stakeholders and strategies

Universe	
Stakeholders	
Technical Strategies	

Phase II - Descriptive Value Analysis

What Values have to be considered, moral and not moral?

Universe	
Stakeholders	

Phase III - Technical Analysis

What biases have to be considered?

Pre-existing bias	
Technical bias	
Emergent bias	

Phase IV - Conflict of Values and Goals

What conflicts exist between different values and goals?

Conflict	Description

Pretheoretical Deliberation

weigh up and group facts, define options for action

Phase V - Ethical System Inspection

If Phase IV did not provide a recommendation for course of action

Arguments	Deontologic	Consequentialist
Pro		
Contra		

Theoretical Deliberation

weigh up arguments

Phase VI - Decision

Should the feature be implemented, why?

if the decision is against the implementation, start again at phase II

Phase VII - Technical Feasability

Technical Implementation

How should the feature be implemented?

Implementation of normative decision

Can the decision made be implemented? Are there aspects that can not be implemented? Should the feature be implemented still?

Linked Issues

Link high level issues here, that are related to the implementation of the feature

Phase VIII - Verification

How can the implementation of the decision be tested? What are good test cases

Linked Issues

link issues that relate to the testing of the implementation here

Instructions Survey EDAP

Introduction

In this survey, you will test the extensions to Jira Cloud that have been developed during the Bachelors Thesis “EDAP- Ethical Deliberation in Agile Processes”. EDAP is a schema that is used to assists teams in the process of documenting the ethical and moral dimension of their development decisions clearly and concisely. It has been integrated into Jira.

Jira has been extended in two ways:

1. The EDAP schema, which is a document that developers can fill out, has been transformed into a Confluence page template.
2. An app has been added that provides a checklist. This checklist app can be filled out before using the EDAP template to provide knowledge and important pointers as to what should be considered during the EDAP process and development in general.

Jira

Jira can be used to manage and document issues (which can be tasks, user stories etc.) in a way that is suitable for agile processes. Jira provides documentation via Confluence pages. These pages can be created using templates, that provide structure that can be filled out.

Scenario

You are a member of a development team using Scrum. Your team is working on a work management and communication tool. Right now, you are in the Sprint Planning meeting. You and your colleagues have selected the following issue from the product backlog: “The productivity of employees is calculated with the data provided by the management functions of the system. The results are presented to the team manager” You and your team are not sure about the moral implications of this goal, so you start the EDAP process. (This example has been adapted from a Microsoft teams feature: https://www.theguardian.com/technology/2020/nov/26/microsoft-productivity-score-feature-criticised-workplace-surveillance)

Sprint Planning meeting

In the Sprint Planning Meeting a set of items from the backlog is selected to be worked on during the following sprint. The selected items are discussed, refined and broken down into more concrete tasks.

Instructions

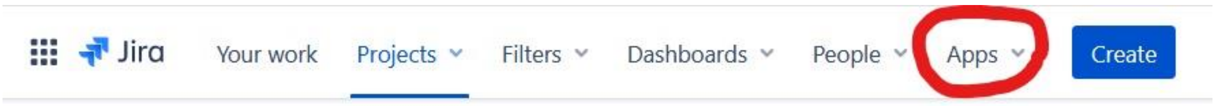
Step 1:

Open https://edap.atlassian.net/ and log in using the credentials:

Email: edaptesting@gmail.com

Password: edaptester

Step 2:



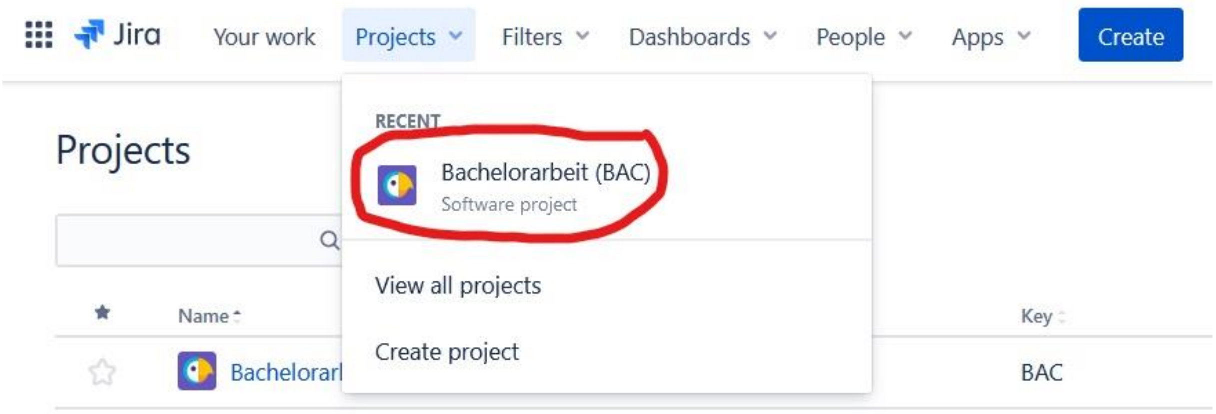
On the top bar, click "Apps", then select "Checklist", you will arrive at the checklist app.

Check "Data related checkpoints", then check all the points that apply to the feature. Then click submit.

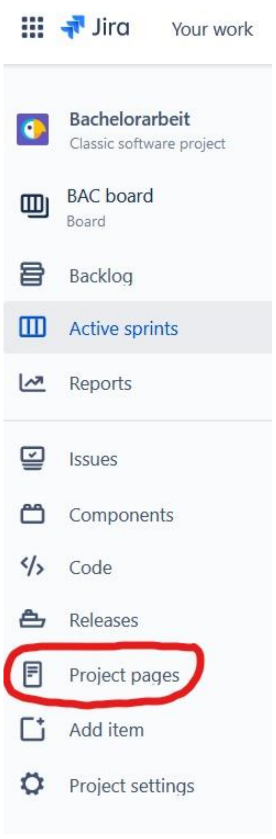
You will be presented with text that informs you about important information that fit the checkpoints you selected. Read the text.

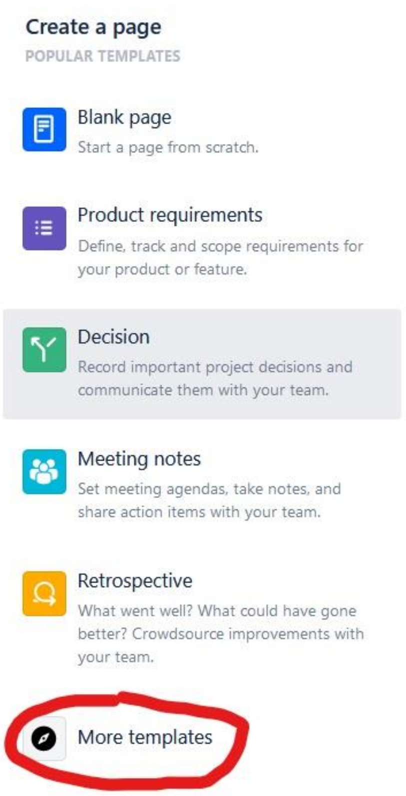
Step 3:

Go to "Projects" on the top bar, then select "Bachelorarbeit".



On the left, click "Project pages". Then, on the right, click "More templates".





Create a page

POPULAR TEMPLATES

Blank page
Start a page from scratch.

Product requirements
Define, track and scope requirements for your product or feature.

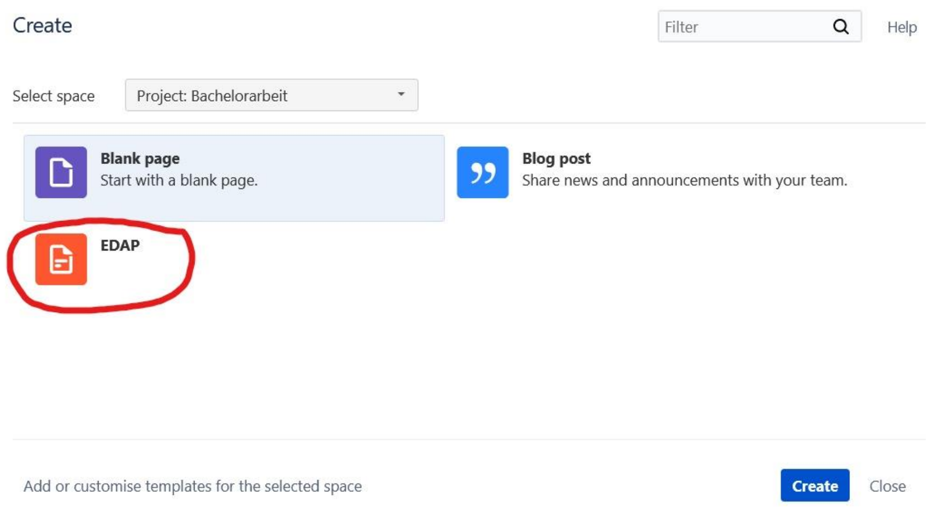
Decision
Record important project decisions and communicate them with your team.

Meeting notes
Set meeting agendas, take notes, and share action items with your team.

Retrospective
What went well? What could have gone better? Crowdsource improvements with your team.

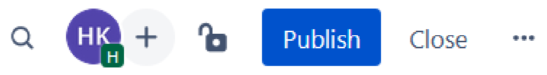
More templates

A new page will open. Select "EDAP" and click "Create" on the bottom. You have now created a page using the EDAP template.



Keeping in mind the information from the checklist, fill out the template. You do not need to be very detailed, focus on documenting all important points in short sentences or bullet points.

If you want, you can save the page you have created by clicking "publish" on the top right. Make sure you have titled the page, otherwise publish is disabled.



Step 4:

Fill out the survey here:

https://forms.gle/XZGwDFYYtM9gJtxY9

List of Figures

1. RE activities in Scrum, [8] 11
2. Example of FF-Tree, [9] 16
3. EDAP process in Scrum, scrum.org with added elements 19
4. Simplified example of EDAP, using https://thenounproject.com/term/backlog/ 20
5. Questions on general experience 26
6. Questions about the checklist app 27
7. Questions about EDAP 28

List of Tables

1. Ratings for agile tools, xx - fully satisfied, x - partly satisfied 13

Bibliography

[1] *11 Best Tools for Creating Scrum Boards Online*. Oct. 2020.

[2] D. 3. *12 Best Scrum Tools for Project Management in 2021 (Jira, etc.)* Feb. 2021.

[3] P. Abrahamsson, O. Salo, J. Ronkainen, and J. Warsta. "Agile software development methods: Review and analysis." In: *arXiv preprint arXiv:1709.08439* (2017).

[4] Benaston. *10 Best Agile Tools For Managing Projects In 2021*. Feb. 2021.

[5] T. Bishop. "Microsoft patents tech to score meetings using body language, facial expressions, other data." In: *Geekwire* (Nov. 2020).

[6] D. N. Chin. "Empirical evaluation of user models and user-adapted systems." In: *User modeling and user-adapted interaction* 11.1 (2001), pp. 181–194.

[7] *Data Ethics Canvas*. May 2019.

[8] A. De Lucia and A. Qusef. "Requirements engineering in agile software development." In: *Journal of emerging technologies in web intelligence* 2.3 (2010), pp. 212–220.

[9] G. Gigerenzer and W. Gaissmaier. "Heuristic decision making." In: *Annual review of psychology* 62 (2011), pp. 451–482.

[10] G. Gigerenzer and D. G. Goldstein. "Reasoning the fast and frugal way: models of bounded rationality." In: *Psychological review* 103.4 (1996), p. 650.

[11] D. Gotterbam et al. "Reducing software failures: Addressing the ethical risks of the software development lifecycle." In: *Australasian Journal of Information Systems* 9.2 (2002).

[12] A. Hern. "Microsoft productivity score feature criticised as workplace surveillance." In: *The Guardian* (Nov. 2020).

[13] M. Hutson and J. Seabrook. *Who Should Stop Unethical A.I.?* Feb. 2021.

[14] M. Kleemans, S. Daalmans, I. Carbaat, and D. Anschütz. "Picture perfect: The direct effect of manipulated Instagram photos on body image in adolescent girls." In: *Media Psychology* 21.1 (2018), pp. 93–110.

[15] B. Laugwitz, T. Held, and M. Schrepp. "Construction and evaluation of a user experience questionnaire." In: *Symposium of the Austrian HCI and usability engineering group*. Springer. 2008, pp. 63–76.

[16] N. K. Malhotra. "Questionnaire design and scale development." In: *The handbook of marketing research: Uses, misuses, and future advances* (2006), pp. 83–94.

[17] N. Morpus. *8 of the Best Agile Tools for Project Managers*. Aug. 2020.

[18] J. Nielsen. "Usability inspection methods." In: *Conference companion on Human factors in computing systems*. 1994, pp. 413–414.

[19] K. Roose. "The making of a YouTube radical." In: *The New York Times* 8 (2019).

[20] A. Sillitti and G. Succi. "Requirements engineering for agile methods." In: *Engineering and Managing Software Requirements*. Springer, 2005, pp. 309–326.

[21] M. Singh and R. Vyas. "Requirements Volatility in software development process." In: *International Journal of Soft Computing* 2.4 (2012), pp. 259–264.

[22] *The 10 Best Agile Project Management Tools of 2021*. Mar. 2021.

[23] *top 10 scrum tools in 2021 Archives*.

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

[25] S. Wagner, D. M. Fernández, M. Felderer, A. Vetrò, M. Kalinowski, R. Wieringa, D. Pfahl, T. Conte, M.-T. Christiansson, D. Greer, et al. "Status quo in requirements engineering: A theory and a global family of surveys." In: *ACM Transactions on Software Engineering and Methodology (TOSEM)* 28.2 (2019), pp. 1–48.

[26] F. Wilson. *Best Agile Tools for Project Management in 2020 - DZone Agile*. Aug. 2020.

[27] N. Zuber, S. Kacianka, A. Pretschner, and J. Nida-Rümelin. "Ethische Deliberation für agile Softwareprozesse: EDAP-Schema." In: ().