Titel

Autor

***Abstract*—Product development is facing new challenges as a result of the demand for increasingly individualized products in small batch sizes and short time to markets. By providing software tools, digitalization is an enabler for agile product development. Stakeholders such as the customer, the project manager, the requirement manager and the solution manager can develop the product together and react quickly to each other's demands. The individual aspects of product development are often considered separately. Each stakeholder uses its own platform to generate outputs. As a result, the connection between the components of product development is missing. Without a common context between the different departments, valuable information and work results are lost. This complicates aspects such as scheduling, priority planning or the definition of new requirements as there is no common basis of information. To solve these issues, we have developed an open source platform on which all relevant results can be managed. Our platform combines project management, requirements engineering and solution engineering to offer a transparent insight into the progress of product development. For the development of the software, tools from web development were used to create a cross-platform user experience built on a client server architecture. The focus was on making the software lightweight and intuitive to use. The software was continuously tested during the development process and showed potential to add value to the project management of products. Nevertheless, important adjustments still need to be made to create a solid user experience. This article describes the development of such a platform and evaluates its benefits, drawbacks and what lessons we have learned. In the Future the software will be further developed in follow-up projects to make it ready for practical use and to extend its functions. There are already plans to extend the software in the direction of a simulation tool and virtual reality.**

I. Introduction

Context

Every engineering design process starts with defining the needs to be covered in the resulting product. In order to successfully develop the product, it is important to capture the customer's needs precisely and effectively [7]. The trend is for products to become increasingly complex and shorter time to markets. To meet today's challenges, companies must shorten the cycle time of new products, reduce the resources used, increase the quality of the product. To meet this business goals, companies need to collect, analyze and apply requirements extensively and effectively [11]. Requirement engineering plays an important role as a discipline and holds many challenges. Clear requirements and a focus on the end user with management support represent half of the success of future product development [1]. Through proper requirement management, a product with higher quality can be developed in less time by meeting the right requirements because fewer iterations are required. The customer satisfaction is increased and through a clean documentation it is possible to reuse parts of the process for other products [2]. By collecting the information and by possible changes of the customer requirements, a lot of data comes up, which makes it difficult to keep an overview of the product development process. It is difficult to manage the large amount of data because of the cognitive capacity of humans. Project management tools help to get an overview of the project progress with visualizations in the form of dashboards [10]. Such a tool must take into account the dynamics of the requirements in the development process and provide support for collaborative and interdisciplinary cooperation. It should be possible to divide the engineering activities into sprints [7].

Problem

The development of complex and sustainable products produces many challenges for requirement management. Requirement management is an effort where data from different stakeholders merge. Changing requirements in one domain also affects other domains. Many tools already exist to support requirement management in the different domains. Due to the distribution of tools, stakeholders often have difficulty defining priorities during the product development phase because they do not have an overview of the big picture. Commercial tools can mostly be used to support requirement engineering and the project management process. If these tools are separate applications and follow different processes, merging the data becomes a major challenge. The division into different systems leads to differences in terminology and concepts, which causes communication problems [9] . The relationship between effort and benefit in a software tool must also be considered. It must be taken into account what maximum effort the stakeholders are willing to invest to use the software. In a tool that is not comprehensibly structured and difficult to learn, not much effort will be invested, and no added value can be generated [10]. Even with a tool that covers requirements engineering and project management, many work results are lost because not all aspects of product development are included. Because CAD data and documentation are often considered separately from management, the context is lost. Requirement engineering task planning and scheduling are therefore difficult. This lack of transparency in product development makes it hard to implement agile product management. If the customer has no access to the development progress, there is no possibility of prioritizing times and requirements.

Solution approach

Since tools to support product development do not cover all essential aspects such as project management, requirement engineering and solution engineering, we want to describe the development of such software and evaluate it in the course of this article. The software considers product development as an

overall concept. This gives the stakeholders a transparent and traceable overview of the project progress and enables agile product development. The platform offers user management, member management project management and provides a 3D view for the uploaded CAD models. On our platform, users can create different products. For each product, members, versions, issues, comments and milestones can then be added and edited. As in GitHub, different versions can be created. Each version of the product contain a 3D CAD model that can be viewed like in a 3D CAD software. Members can be registered for a product which possess different permissions. With the help of the customer member role, the customer can participate in agile project management and track the current status of product development. Issues, Comments and Milestones serve as project management tools. Milestones are filled with issues that can be open or closed. A milestone is considered complete when all associated issues are closed. Each issue contains a communication channel where the respective task can be evaluated. On the platform, the CAD model can be viewed in a 3D view and is connected to the integrated product management tools.

Outline

This article is structured as follows: In Section II we discuss related work on the problem defined previously. In Section III we present our original solution to the problem at hand. Then, in Section IV we evaluate our solution with respect to different criteria. Finally, in Section V we summarize our learnings and provide an outlook on future work.

II. Related work

Scientific approach

Google Scholar was used to search for appropriate literature using the following search terms in various combinations: Requirements Engineering, requirements management, computer aided design, product development, mechanical engineering, mechanical design, industrial design, traceability, integration, product lifecycle management, software tools, agile project management. For each combination searched, the first 30 entries were looked through, and the most relevant articles were used as state-of-the-art references. Most of the articles focus on requirement management and describe its theoretical aspects. These articles also point out the importance of well-executed requirement management and the challenges that occur in the process. It is mentioned again and again that requirement engineering is a major part of success. We found articles that divide requirement management into subcategories and phases, describing each aspect of developing new products in a high-tech company [11]. In product development, it is important to be able to reuse the knowledge generated. One article described a framework for combining requirements management with engineering design and making it reusable. The resulting design reuse system consists of three key elements: process knowledge, task knowledge and product knowledge [2]. An article provides a scenario based approach for the identification, elaboration and specification of engineering design requirements using a three-phase model [7]. Another article provides a literature review to show how requirements engineering can be further enhanced with the help of guidelines for requirement management process improvement [5]. Computer aided software engineering tools help to support the development process. In one article such a CASE tool was developed as an Eclipse plugin which enables traceability and accessibility from the planning to the coding phase. This work focuses on the development of software products [4]. An article shows the importance for requirements engineering to use visualization tools to help stakeholders catch data. They describe different elements for visualization and want to implement a software tool based on their results later [10]. An article points out what methods and tools are important for combining requirements engineering with project management and how important it is to see the two fields as an overall concept. The challenges of requirements engineering and project management in different contexts are highlighted. A framework is developed that combines requirements management and project management to increase the quality and effectiveness of multidomain development processes [9]. Another contribution is the implementation of a tool for model based requirement engineering for agile product development. The concept of the tool is to connect the product requirement with the development task and the corresponding test case. Based on this a dashboard was built which received positive feedback and will be further developed [12]. In addition, a tool was built using web development tools such as Angular.js and Node.js to support product development. The tool supports the business, operational and technical aspects for service-oriented software [3]. Another paper compared different software tools for enhanced demand compliant design based on different criteria. The analysis shows how important it is to use the right tool and that most tools focus only on the own discipline and rather a universal and transdisciplinary approach should be chosen [8]. In one article, the importance of requirement engineering is discussed and attention is drawn to the fact that most studies and research work that is available has many strategies but no real implementation [6].

Commercial approach

GitHub: It is a service for version management of software projects. Repositories are uploaded and managed here. Multiple versions can be uploaded from the repository and tasks can be distributed, divided into milestones and discussed in the issues section. These functionalities solve some problems defined above. However, GitHub is only designed for software products and does not support CAD files. Thus, solution engineering of mechanical systems cannot be included. We find GitHub to be an excellent platform for software development and in the course of this article we want to create a platform that offers similar tools, but is suitable for 3D CAD models [1].

Onshape: The tool onshape is a premium software, from the company PTC, which combines product development, the CAD, data management, collaboration and real-time analysis. The tool offers a huge range of features. It provides a secure cloud workspace for all project stakeholders and

[1] https://github.com/

the development team can work together on the design of the product. The tool enables multiple parallel design iterations and real-time collaborative work, and offers many more features. Unfortunately, the software also has disadvantages compared to lightweight open source software due to the huge scope of functions and the binding to one company. If the customer is involved in an agile project development process, he has a high entry hurdle when learning the program. The customer can get lost in the tool and the actual aspects of the product to be developed can be lost in the process. In order to keep the tool consistent with real project progress, it must be constantly updated. Due to the high amount of data in the tool, the stakeholders and especially the customer can lose the overview. For the customer, it is easier to see exports of product versions than the entire system when making decisions, and the company does not have to disclose design details. The deep integration of Onshape into the world of PTC forces you to be caught in it. Here we want to go the approach of an open source software that works with 3D CAD exports. Furthermore, the software should be intuitive to use and easy to learn. Because the entire development process does not take place in our tool, but only exports are uploaded, there is also freedom with which data the platform is fed and what the customer gets to see. The customer should be able to participate in the product development on the basis of the product versions provided without having to see complex design details [2].

Conclusion: Most of the papers we have found through research highlight the area of requirements engineering from the product development process in particular. They provide many theoretical concepts to point out the importance, challenges and methods for improvement. Only a few articles describe a concrete implementation, but these only cover requirements engineering and not the entire product development process. The two commercial solutions GitHub and Onshape are excellent tools in their domain but in our opinion are not suitable to support the agile product development process with close customer collaboration. In the case of GitHub, it's because it doesn't support design drawings and oneshape software is an expert tool that delivers a huge feature set, which can make the customer lose track by access all the design details of the product. After our research we came to the conclusion that our field of research is new and important and the market is by far not yet saturated with tools that provide lightweight support for product development.

III. Our solution

For the implementation of the project, the requirements for the software are defined first. The requirements are still general in this first section and become more concrete during the development of the design. Following the requirement definitions a detailed solution is compiled step by step. The most important concepts of this software are thereby the software architecture, the data model, the function model, the permission model and the user interface. Thus, the software can be developed according to the defined concepts and evaluated afterwards.

[2]https://www.onshape.com/

Conception

The goal to be achieved is to combine requirement engineering, project management and solution engineering with the corresponding CAD data to create a compact and lightweight tool to support product development. Customers and companies can evaluate and control the development status on a common basis. For this purpose, the platform GitHub was considered as a reference, which already offers a similar solution in the area of software development. To involve the customer in the development process clear user roles with corresponding permissions must be defined. For example, the app needs a user administration with an associated login function. At the top level, it must be possible to define which users are allowed to create products and edit the list of users. The person who creates a product is both owner and manager and can add product members who have different permissions. Depending on the distributed rights, members can add versions to the respective product. Each added version contains a new CAD model with version number, previous version and description. For each product it should be possible to create issues, which are filtered by open and closed issues. In each issue there is a communication channel in which the issues can be discussed. Furthermore, specific components of the model can be selected and referenced, and an issue can be closed or reopened. Another level of product management is built by the milestones. Milestones can be created, and different issues can be added to them. A list of open and closed issues as well as a chart show the progress of each milestone. Finally, the platform must provide settings for each product to define the member list and the product properties. All data generated on the platform should be stored in a database. The software should be as lightweight as possible and intuitive to use. The customer should be in the foreground and the user interface should be built from the user's point of view. In this way, a platform can be created that clearly combines product development, product management and communication with the customer.

Architecture

In the first step of the conception the architecture of the software is planned. The following picture [see Fig. 1 on page 4] shows the general structure of the software. It is a full stack app written in Typescript and consists of gateway, frontend, backend, common and the database at its core.

Gateway: The gateway runs at port 3000. It establishes a connection between the services backend, broker, worker and frontend. When the app is started the gateway serves as the entry address. It displays the frontend, which itself runs on port 3003, while the backend is running in parallel.

Frontend: The frontend provides an interface to make changes to the data via the offered functions of the developed API. To create the user interface React version 17 was used. It uses React states that can be altered with React hooks. The frontend communicates via HTTP requests with the backend to create, read, update and delete data with help of the API located in the backend. This communication runs with the library Axios. To display the uploaded cad models three.js

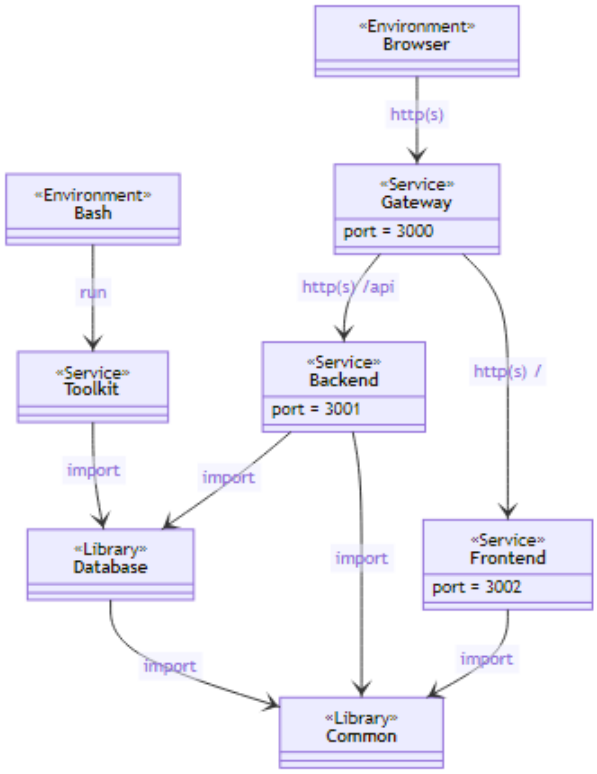

Fig. 1. Packages

was used. It offers a 3D view of the selected CAD file. The view of the model can be adjusted by zooming and rotating, as well as positioning. The library Recharts provides a variety of different charts and graphs. Recharts is used to display a burn down chart in a specific user interface component.

Backend: The backend of the app is built with the framework nest.js. This area of the software is the core component and provides those functions to meet the previously defined requirement specifications. Using nest.js, an API was created that interacts with the data using CRUD commands and sends it to the frontend. For that, it receives commands from the frontend and then executes the corresponding function of the API. The backend is controlled via HTTP requests. Therefore, the commands can be executed without the frontend. Good possibilities are offered by the program Postman or with Swagger. A Swagger integration was done in the backend during the development process. Swagger provides good documentation about the API and allows its functions to be executed without the frontend.

Common: The package common provides the necessary interfaces for the communication of the frontend and backend. In this section the data classes like product, version, issue, etc. for the different functions are defined. The rest file contains the interfaces for the HTTP requests. They define exactly which data may be sent from the frontend and which may be received again. For this purpose, the CRUD methods with transfer parameter and return value are defined in each interface of a data class.

Database: To store the data permanently a PostgreSQL database is used. It runs in a Docker container on port 5432. The database uses the data classes defined in the interface as entities. For the object relational mapping TypeORM is used.

Toolkit: The toolkit package provides functionality to fill the database with test data. Based on the classes defined in the interface, the respective objects are instantiated here. These objects are filled with fictitious data to test the functionality of the app. After the objects are instantiated, they are loaded into the database. For this purpose asynchronous function calls are executed.

Data model

The core of the software is in the backend. Via the API, the data can be manipulated using CRUD methods. This section describes the data model of the app. The necessary classes and entities are based on the interfaces in the common package. The difference between the data in the backend and in the database is that each entity class in the backend is divided into update, add and the base class. In the database, the classes are not divided. For example, the class product is mapped to a table in the database.

Entities: The data model contains following entities [see Fig. 2 on page 6]: User, Product, Version, Issue, Comment, Milestone, Member. At the top level of the data model is the user, who can create multiple products if he has the permission to do so. For each product versions, issues, comments, milestones and members can be created. The entity User has personal attributes like name, email, password and the two permissions to create and change users or to create products. Finally, like any other entity, the user has a deleted flag. If a delete method is executed, this is set to true. The find and get methods of the API are implemented to search only for objects that are not deleted. When deleting, care must be taken which dependencies must also be set to deleted. If a user is deleted, no further data is removed from him. The user is only marked as deleted on the user interface. All data generated by the user will remain. All other entities are linked to the product entity. The product itself does not provide much information, because it will be filled with information later by other references. If a product is deleted, all versions, issues, comments, milestones, and members must also be set to deleted. The versions contain important information and the 3D CAD model. To create a versioning, the attributes major, minor and patch are specified for a version. By knowing the current version and previous version, a graphical overview can be built on the UI. Multiple issues can be added for each project. The state property distinguishes between open and closed issues. So it is possible to filter for completed and not completed issues. For each issue one or more comments can be created. With the attribute action an issue can be closed or reopened by the comment. For each product one or more milestones can be defined. These have a label and clear start and end dates. Later it can be counted how many issues per milestone are still open or already closed. This can be used to display a graphical representation of the project progress in the frontend. For each product one or more members can

be added. The attribute role can have three states: manager, engineer, customer. Depending on the role, the corresponding product member has different permissions to interact with the respective product.

Function model

The functions implemented in the API allows to create, read, update and delete data. This would not even require a graphical user interface. All functions offered by the API are called via HTTP requests. These requests can also be made via the console or other software such as Postman or Swagger. Thus, the functional model in combination with the data model is the foundation of this application. The API is built with nest.js. One rest entity contains of controller, service and module. In operation, an HTTP request is performed in the frontend. The data flows from the respective view via a request manager to a rest method, where the respective Axios request is triggered. In the backend, the nest controller receives the request and sends it on to the nest service. The nest service implements the necessary API methods to make changes to the data. These functions are all asynchronous and have a promise as return value. After processing, the result is returned. This tunnel is provided with various permission checks to verify that the user or product member has the permission to perform this action. The interface in the common package controls the communication between frontend and backend. In the nest service the methods find, add, get, update, delete and convert are available.

User management: Users can be created, modified and deleted here. The condition for this is that the respective user has the User Manager Permission. When a user is created, the associated data object is filled with the respective attributes. Permissions can be assigned to allow the new user to create and edit other users and to create products. For this the two attributes user management permission (ump) and product management permission (pmp) have been added. The API offers the following methods for interaction with user objects:

- findUsers(query?: string, productId?: string)
- addUser(email: string, name: string, passwort: string, ump: boolean, pmp: boolean, file?: File)
- getUser(id: string)
- updateUser(id: string, email: string, name: string, passwort: string, ump: boolean, pmp: boolean, file?: File)
- deleteUser(id: string)

The `findUsers` method searches for all users of the system from the database. It can be searched either via a query by username or via the product id. The parameters query and productId are optional. Depending on which parameter is specified, the search process runs differently. For the `addUser` method the transfer parameter is needed, which contains the information about the new user. The second parameter holds the profile picture and is optional. `GetUser` searches for a specific user based on the userId and returns the object. `UpdateUser` needs the userId, the new parameters and optionally a file containing a profile picture. `DeleteUser` sets the deleted flag of the particular user to true. The get and find methods are implemented that way, to search for objects that are not deleted.

Product management: This part of the API provides methods to make modifications to product objects. Here products can be created, modified and deleted. The following bullet points show the methods that are available for interacting with products:

- findProducts()
- addProduct(userId: string, name: string, description: string)
- getProduct(id: string)
- updateProduct(id: string, name: string, description: string)
- deleteProduct(id: string)

The `findProducts` method searches in the database for all products that are not deleted. For this no passing parameter is necessary. The other methods work according to the same principle as in user management. It is searched with the corresponding productId for the object and displayed, or its properties get changed. When deleting a product, it must be ensured that all objects that are attached to a product are also deleted. So if a product is deleted, all versions, issues, milestones and members must also be set to deleted.

Version management: Only by referencing a version to a product it gets further descriptive properties and the 3D CAD model. Again, the API provides CRUD methods to interact with versions:

- findVersions(productId: string)
- addVersion(userId: string, productId: string, baseVersionIds: string[], time: timestamp, major: number, minor: number, patch: number, description: string, file: File)
- getVersion(id: string)
- updateVersion(id: string, major: number, minor: number, patch: number, description: string, file?: File)
- deleteVersion(id: string)

Unlike user management, a file must be specified when a version is created. This is an STL file, which is then stored locally. This CAD model can be exchanged when updating the version.

Issue management: These methods are available to create, read, update and delete issues:

- findIssues(productId: string, milestoneId?: string, state?: 'open' or 'closed')
- addIssue(userId: string, productId: string, time: timestamp, label: string, text: string, state: 'open' — 'closed', assigneeIds: string[], milestoneId?: string)
- getIssue(id: string)
- updateIssue(id: string, time: timestamp, label: string, text: string, state: 'open' — 'closed', assigneeIds: string[], milestoneId?: string)
- deleteIssue(id: string)

The `findIssues` method can search for issues in the database in several ways depending on the transfer parameters. An issue is always connected to a product. The state parameter allows to filter the issues by closed and open. This is especially useful to be able to filter for completed or uncompleted tasks for the purpose of product management. When deleting issues, all comments belonging to the respective issue must also be deleted.

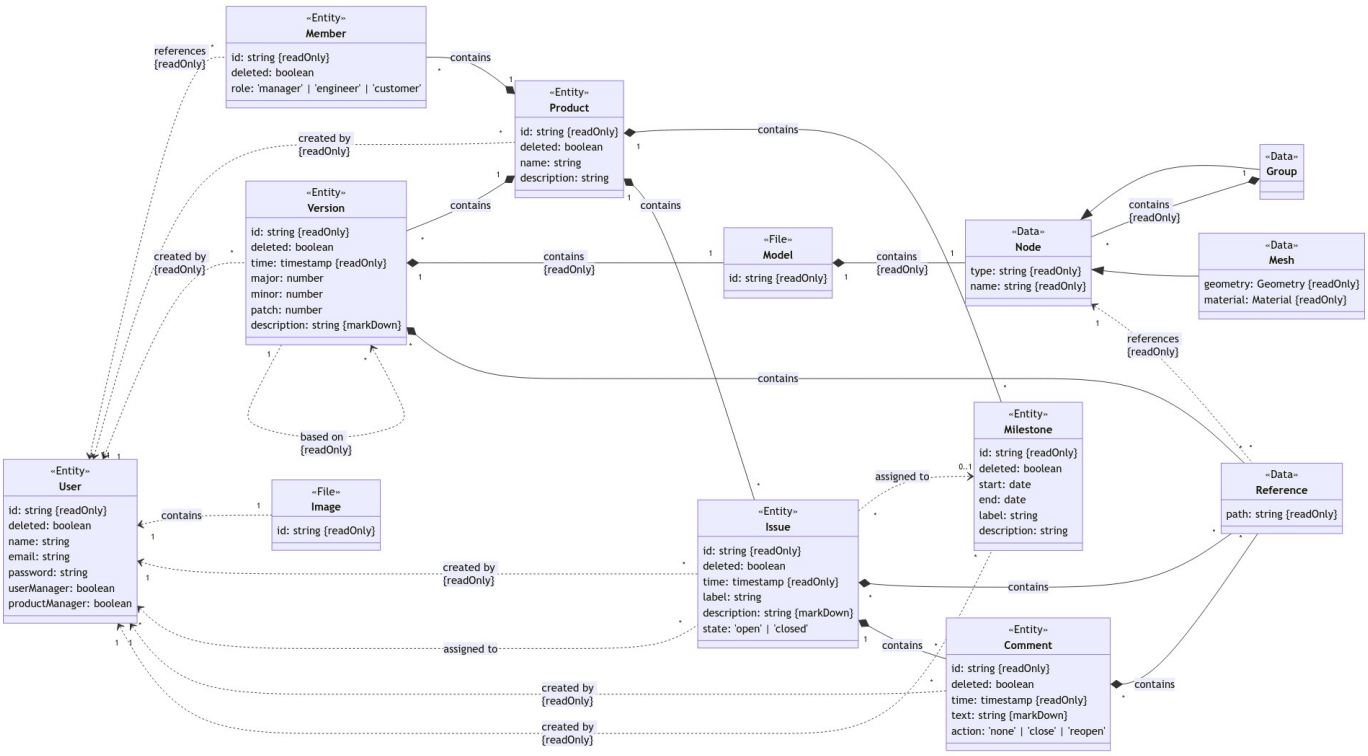

Fig. 2. Data model

Comment management: The following list shows the methods that can be performed to interact with the comment objects:

- findComments(issueId: string)
- addComment(userId: string, issueId: string, time: timestamp, text: string, action: 'none' — 'close' — 'reopen')
- getComment(id: string)
- updateComment(id: string, text: string, action: 'none' — 'close' — 'reopen')
- deleteComment(id: string)

An issue can have one or more comments. These comments are linked to an issue and the issue is bound to a product. The `findComments` method stores all comments for a given issueId returns them.

Milestone management: Milestones are a container that can be filled with issues by reference. The CRUD methods are listed below:

- findMilestones(productId: string)
- addMilestone(userId: string, productId: string, label: string, start: timestamp, end: timestamp)
- getMilestone(id: string)
- updateMilestone(id: string, label: string, start: timestamp, end: timestamp)
- deleteMilestone(id: string)

The methods have analogous purposes as already with the other data objects. When deleting a milestone, it must be ensured that the references to the linked issues are removed.

Member management: The API provides the following methods for member management:

- findMembers(productId: string, userId?: string)
- addMember(productId: string, userId: string, role: 'manager' — 'engineer' — 'customer')
- getMember(id: string)
- updateMember(id: string, role: 'manager' — 'engineer' — 'customer')
- deleteMember(id: string)

Members are always linked to the respective product. Additionally, the userId can be used to find one specific member. Then the `findMembers` method returns an array of members that can then be displayed on the user interface. The other methods are similar to those of the other entities.

Permission model

The permission model offers a number of possible restrictions on the platform so that not every user has all freedoms. This is especially important when cooperating with customers. The customer should only have the possibility to evaluate existing products. For this he can see the products he is registered for and use the given product management functions. Creating new users and products should only be possible by the developers of the respective product. They also organize the rights of each user. The permission model is divided into two levels. The first level is the user level and the second level is created by the product members. This system is deeply integrated in the backend.

User level: The User entity has the two attributes user management permission and product management permission.

Only a user who has the user management permission can create or edit users. On the other hand every user can edit his own personal data. With the product management permission it is possible to create new products. If a new product is created, the respective user is automatically also a product member and receives the member role manager. The permissions of each user can be adjusted afterwards. So it is possible to give multiple users the permissions for user administration and product administration. For example, a user and product manager can exist for each department.

Member level: For permission management at member level, three roles are provided. These roles are: manager, engineer and customer. The manager is the one who created the product and has the permission to add more members. So he can add more managers, who in turn have all the rights over the management of the product. This is useful when the product development covers several departments. The second role is the engineer. He is involved in the product development process. He has no rights to change the product or its members. However, he can freely create and edit versions, issues, comments and milestones. The last role is the customer who has the possibility to observe the product development process. He can follow the progress of the project, but has no permission to change anything on the platform. In the current version of the software the rights of the customer are still very strict. The permission system is implemented in such a way that it can be changed with few adjustments. If the customer needs writing permissions, this can be easily changed.

Interface model

The user interface provides a convenient way to interact with the functional model to modify and display data. The user interface offers a consistent design, which runs through the entire system. The header of the user interface offers three buttons. A click on ProductBoard leads back to the start page. On the right side there is the user administration and the currently logged-in user. The button to view the users is only visible if the current user has the user management permission. The rest of the page below the header adapts to the corresponding content.

Product view and ProductSettings view: After a successful login with username and password you will see the product view. This page lists all available products in a table [see Fig. 3 on page 7]. For each product in the table a preview is shown. The other columns show the attributes Owner, Name, Description, Versions, Issues and Members. The X on the right provides the possibility to delete the corresponding product. The owner is the person who created the product. Name and Description are defined when the product is created and can be changed later in the Product Settings [see Fig. 18 on page 10]. The columns on the right show how many versions exist for this product, how many issues have been created and how many members have access to the product. By clicking on New product you get also to the ProductSettings view where you can add a new product [see Fig. 4 on page 7]. This button is only visible when the corresponding user has product management permission. After entering name and description the new product appears in the product list on the start page. Only by creating a new version for a product a CAD model with further information is added.

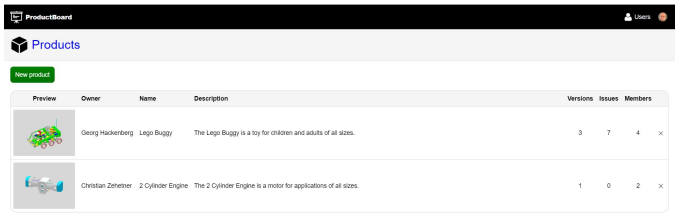

Fig. 3. Product view

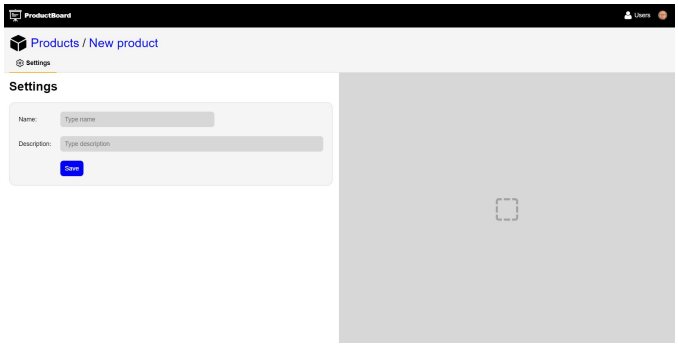

Fig. 4. ProductSettings view

User view and UserSettings view: This view is only accessible if the active user also has the authorization. By clicking on Users this page is called. If the user does not have user management permission, the button is not visible. On this page all users with profile picture, name, email and permissions are displayed [see Fig. 5 on page 7]. You can delete a user by clicking on the X on the right side. If a user is clicked on, he can be edited via the User settings. The button New user also leads to the user settings where you can provide information and create a new user by clicking on the save button [see Fig. 6 on page 8]. Save leads back to the user overview and shows the new or changed user in the table.

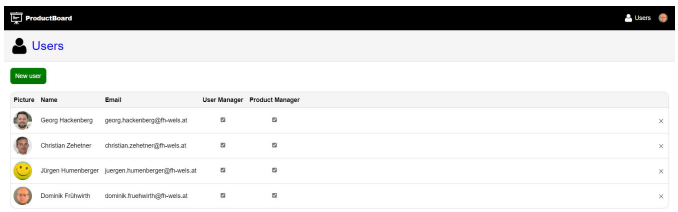

Fig. 5. User view

ProductVersion view and ProductVersionSettings view: Clicking on a product takes you to the ProductVersion view [see Fig. 7 on page 8]. You can also use the toolbar to jump to other pages such as Issues, Milestones Members or Settings. Next to the links a number in brackets shows how many objects per category have already been created. The left side of the Version view shows the created versions. On the left side there is a tree structure similar to Git. In the

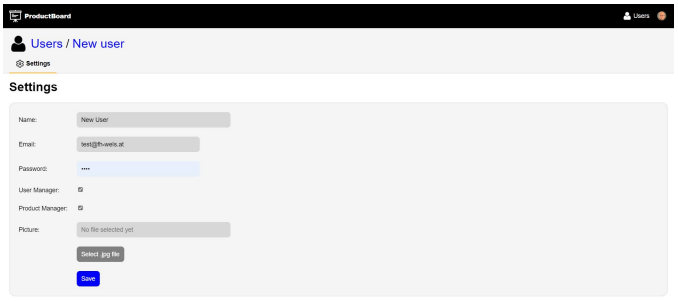

Fig. 6. UserSettings view

middle is the corresponding version number with the owner of the version inclusive email and a short description. Each version offers a preview. By clicking on the respective version, the 3D view on the right side also changes and shows the selected model. The 3D view was built with three.js. It allows to rotate, move and zoom the model. With a click on New version you get to the ProductVersionSettings views [see Fig. 8 on page 8]. There you can enter information for a new version and select an STL file. Depending on the selected base version, the ProductVersion view shows the new version with the corresponding new tree structure after pressing the Save button.

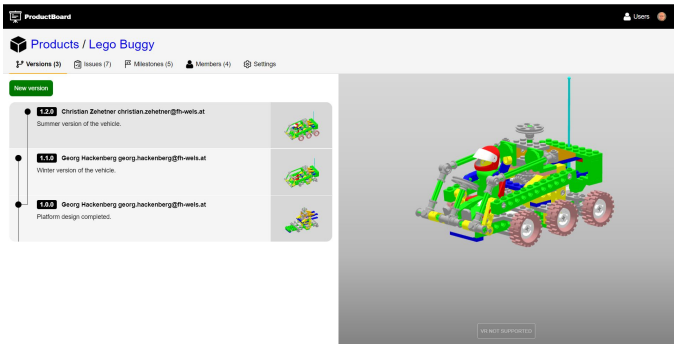

Fig. 7. ProductVersion view

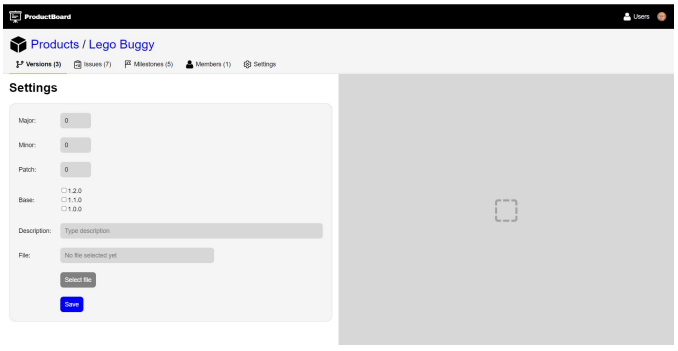

Fig. 8. ProductVersionSettings view

ProductIssue view and ProductIssueSettings view: By clicking on the Issues link, you access the ProductIssue view [see Fig. 9 on page 8]. Here the created issues are displayed in a table. The two buttons Open Issues and Closed Issues can be used to filter the list accordingly. The table shows the reporter who created the issue, the associated label, the assignees and how many comments and marked parts are in the conversation channel. The ProductIssueSettings view allows to create new issues for the product [see Fig. 10 on page 8]. The label, the text, the milestone and the assignees can be defined. An existing milestone can be selected with the dropdown menu. An issue must not be assigned to a milestone. This choice lies by the user. The Save button closes the settings, and you return to the ProductIssue view where the new issue is visible. All views with 3D View offer the possibility to select a desired version for viewing. In the version view the version can be clicked directly. In the other views the version can be selected via a dropdown menu. This menu is located in the upper left corner of the 3D View.

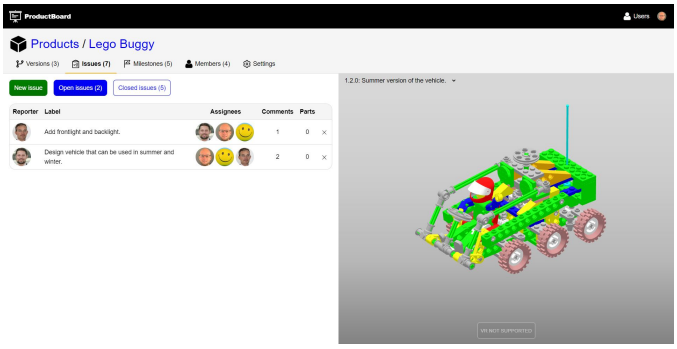

Fig. 9. Issue view

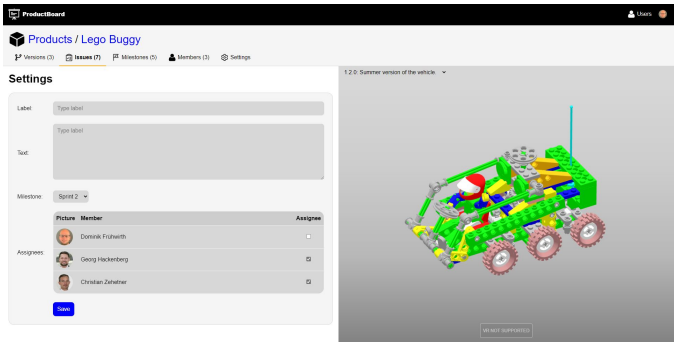

Fig. 10. Issuesettings view

ProductIssueComment view: Clicking on an issue in the Issue view opens the corresponding ProductIssueComment view [see Fig. 11 on page 9]. Here you have the possibility to discuss the issue. You can click on a component of the 3D model in the Comment view to reference a component [see Fig. 12 on page 9]. If a post is created or a part is selected in the comment view, the corresponding counter in the issue view table is increased. A comment can be used to close an issue by clicking Close. This issue will then be found in Closed Issues in the ProductIssue view. With the comment function it is also possible to reopen the issue in the same way. The Close button displays the text Reopen when an issue is closed. In the upper right corner of the ProductIssueComment view there is a button to edit the selected issue. For example, the issue can be assigned to another Milestone or other attributes can

be changed [see Fig. 10 on page 8]. A click on Save makes the changes visible in the ProductIssue view.

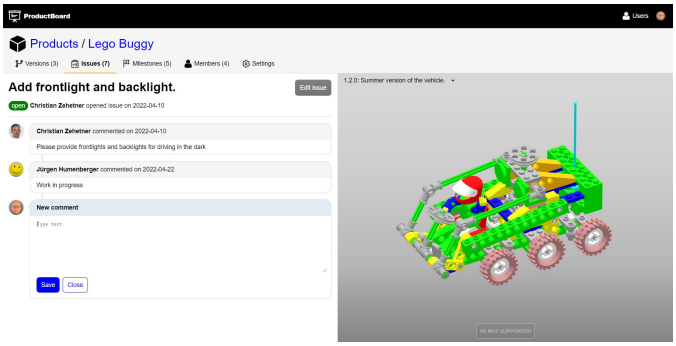

Fig. 11. ProductIssueComment view

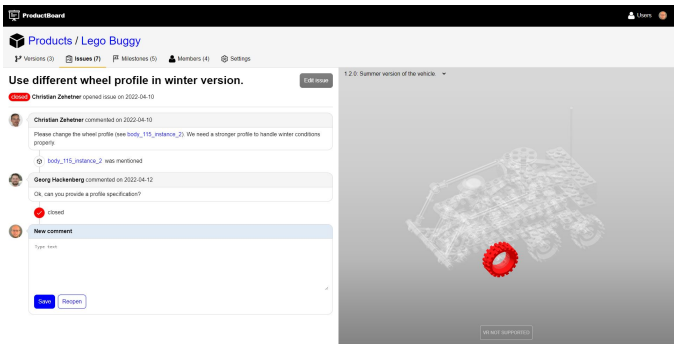

Fig. 12. Selected part in ProductIssueComment view

ProductMilestone view, ProductMilestoneIssue view and ProductMilestoneSettings view: The ProductMilestone view can be accessed via the Milestones link [see Fig. 13 on page 9]. A table shows who created the milestone, its name, start date, end date and the progress like open and closed issues. For each milestone two progress bars are displayed. The first one shows the date progress and the second one the issue progress. If a milestone is selected, a table with the attached issues is displayed [see Fig. 14 on page 9]. This table is identical to the one in the ProductIssue view. Here you can also filter by open and closed issues. On the right side a burn down chart is displayed which shows the current progress of the milestone. The chart shows the start date, the end date, the number of issues and the progress until the current day. A click on New Milestone or Edit Milestone leads to the ProductMilestoneSettings view [see Fig. 15 on page 10]. Here the attributes of a milestone can be adjusted and saved. The new or edited Milestone than show up in the ProductMilestone view.

ProductMember view and ProductMemberSettings view: To distribute the rights for a product, members are added to an existing product via the user interface. In the ProductMember view, a table shows all members who have access to the selected product [see Fig. 16 on page 10]. The table shows the user picture and the name of the user. The column Role defines which rights the respective member has. At the moment there are three roles: manager, engineer, customer. As with every overview table, objects can be deleted from the list by clicking

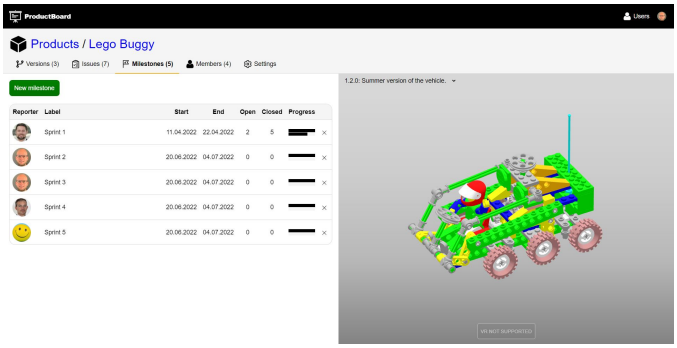

Fig. 13. ProductMilestone view

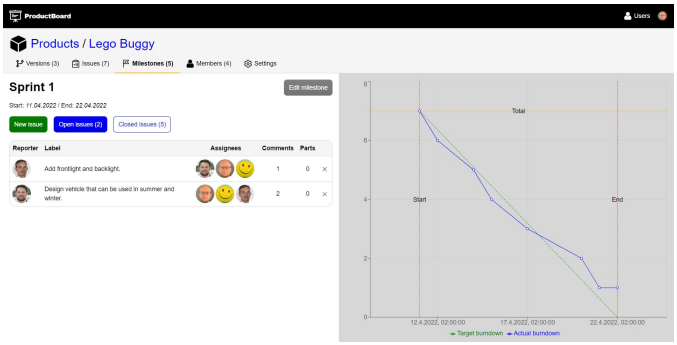

Fig. 14. ProductMilestoneIssue view

on the X button. The button New Member leads to the ProductMemberSettings view. Here you can enter the name of a potential member into a text field. When typing, a list appears in the lower area that filters for each new letter. If you click on a user from the list, this user is selected and a member role can be assigned to him [see Fig. 17 on page 10]. If you click on X, the user disappears and the text field appears again. However, if Save is clicked, the current member is saved as specified and displayed in the ProductMember view.

ProductSettings view: In this view the attributes Name and Description of the selected product can be changed [see Fig. 18 on page 10]. After clicking the Save button the changes are visible in the product overview

IV. Critical Evaluation

To solve the issues defined at the beginning of the article, we have developed the software ProductBoard. This software combines agile project management with product development. It was developed as a team of two people and the development time is currently one year. Due to the short development time, the software has of course not yet reached its final state. Nevertheless, it already covers a wide range of required issues. On the other hand, we have noticed a few points that are still insufficiently covered in the current software version or are still open questions.

Support for multiple file types

At this stage, the tool only supports 3D CAD files in .GLB format. For more interdisciplinary use cases, the software

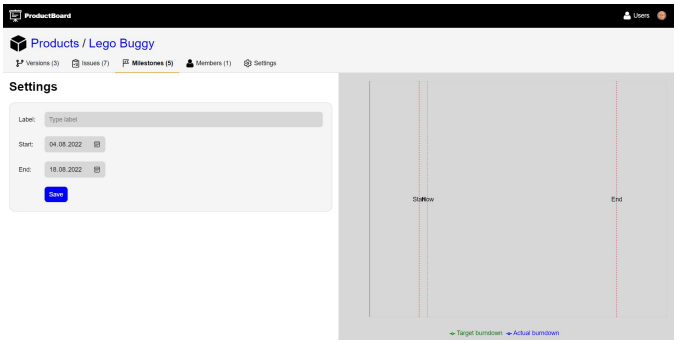

Fig. 15. ProductMilestoneSettings view

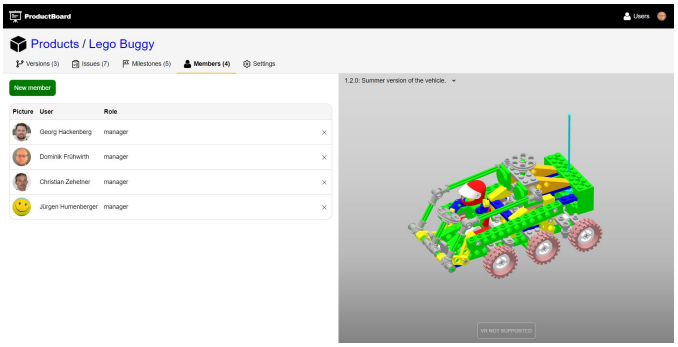

Fig. 16. Member view

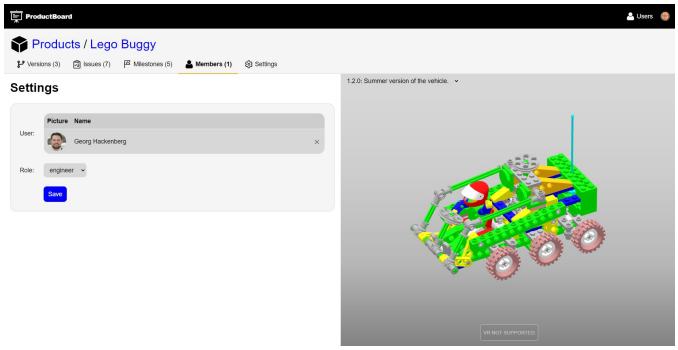

Fig. 17. Membersettings view

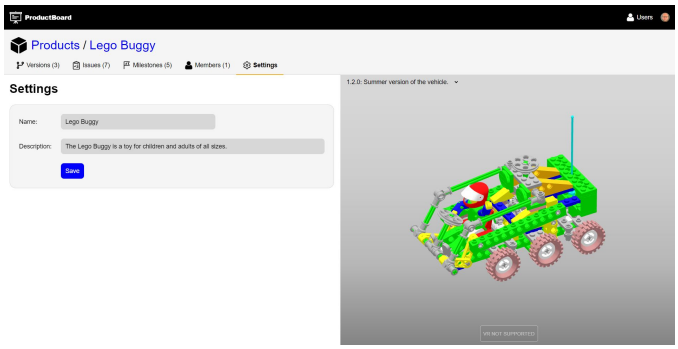

Fig. 18. ProductSettings view

needs to be modified to read and display exports in different formats. The software can also be extended to include image files, video files, or simulations. To do this, the reference mechanism in the data model must be modified so that the node can point to multiple models. For this we have already planned to let the data reference node inherit from other nodes that contain further file formats and will be implemented at a later time.

Bind management tools on products or versions

During the test phase, we asked ourselves whether it is better for mechanical designs to tie the issues, comments and milestones to the respective product or to a product version. Both approaches have different advantages and disadvantages. Like the GitHub platform, our implementation binds the issues, comments and milestones to the respective product. This has the advantage that there is a better overall view of the product and the product can be managed version-independently. The disadvantage is that it can happen with the comments that components are referenced that no longer exist in the current product progress. The conversion, that the issues, comments and milestones are linked to the version would require the entire software to be rebuilt and is not planned after discussions.

Referencing components across multiple versions

In the comments section, different components of the CAD model can be referenced in text and get highlighted. The problem is that it is possible that in the course of a discussion, components of different versions are referenced because the product changes in the course of the project. In the communication channel, however, only one version of the product is visible and thus only the components that have been referenced in this version can be seen. In the test phase it turned out that this is not a good solution. Here an overview must be built in the user interface, which can display all versions with the respective references so that the progress of the product is traceable.

3D view loads always the latest version

If you switch between the different views in the user interface, for example from the issues to the milestones, the current product version is always loaded and displayed in the 3D view on the right side. Therefore, in each view the version to be displayed must again be selected separately via the dropdown menu. Here the user interface must be modified in such a way that the last loaded version is displayed in each view in order to increase the user experience.

Rigid permission model cause problems

The primitive role system that is currently implemented at the member level is not suitable for complex organizational structures. Here, the permission model and the data model must be modified so that the permissions can be fine-tuned. We see either the approach with several fixed member roles or with single permissions, which can be assigned to the respective member. In the user interface, the respective area must then also be designed so that the rights can be distributed more

individually. In our opinion, the permissions on the user level are sufficient to be able to create users and products because the fine-tuning of the permissions then takes place via the member level.

Tool is not optimized for mobile devices

The platform is currently designed exclusively for PC screens and not intended as a mobile application, which is a major weakness. For this purpose, the user interface must be changed to a responsive design to look good on all devices. If the software becomes popular in the world of product development, a change to a responsive design will be necessary. For this purpose the user interface will have to be adapted to all possible end devices.

Asynchronous versus Synchronous collaboration

Currently, asynchronous collaboration like on GitHub is possible using the tool. Setting up a synchronous collaboration within a company structure including the customer would bring many advantages, disadvantages and development effort with it. For synchronous collaboration, for example, care would have to be taken to ensure that user experience does not suffer if new content is constantly popping up at any point. Currently, the software is designed for asynchronous collaboration and whether a change will take place will be evaluated in the future.

V. Conclusion

Summary

The topic of product development and especially requirement engineering is very well covered by literature. Most articles deal with theoretical concepts on how product development can be improved. According to our literature research there are many excellent concepts but only few software tools that solve the problems practically. Thus, we have developed a software tool that covers aspects such as requirement engineering, project management and solution engineering to provide a coherent overview of product development. All relevant work results are clearly visible on one platform and can be managed. The software is lightweight, easy to learn. Exports from 3D CAD models can be uploaded and managed with our version management. With built-in tools such as Issues, Comments and Milestones, agile product design can be performed with the customer. We have tested and critically evaluated our software. Thereby a few questions and issues came up which we discussed for further improvements.

Outlook

In the next steps, the issues from the critical evaluation will be addressed and improved. Furthermore, we have found out some more points to extend the platform practically. The following points will be discussed and implemented in the near future.

Support for multiple file types

The software will be extended in the future so that different data exports can be uploaded. The view on the right side of the user interface can be used for all kinds of data. For example, we would like to make it possible to display video files, images and simulations like CFD of FEM directly in the tool. Videos and simulations of systems could be integrated directly into the tool and controlled there. Further media would be images or PDF files as well as two-dimensional construction drawings or electrical plans.

Test cases for requirements

The requirement management is implemented in our tool via the issues and comments and enables the requirements to be presented in text form. An extension for the requirement management would be the implementation of automated test cases. By formalizing the requirements, test cases can be generated that automatically check for the fulfillment of the issues. Thus, the issues can be automatically closed or reopened depending on the test results.

Linking issues

It can happen that two or more stakeholders create the same issues with a different wording. These issues then have similar or the same requirements, lie around on the platform and reduce the overview of the progress. One improvement is to make the issues linkable to each other. This way, duplicates can be found and removed by linking them.

Extract and display version deltas

Another extension we would like to work on is calculating the deltas between the versions. So the user can see what has changed from one version to the other. The question is whether the delta is supplied by the cad program or whether it has to be determined explicitly. If the delta has to be determined, an algorithm can be implemented that calculates the differences between the CAD data of the versions. Another approach would be that the user manually describes the differences when adding a version. However, this is a pragmatic solution and does not provide a good user experience. The user interface can then be adapted so that the version differences are clearly displayed.

Virtual Reality

Virtual reality has a huge potential to display the CAD models even better. Through VR, the communication between the stack holders can be further improved because the work results can be better viewed. Furthermore, the customer has the possibility to freely view complex components in the virtual environment and thus to better follow the product development and to better evaluate the progress.

REFERENCES

[1] PC Anitha and Beena Prabhu. Integrating requirements engineering and user experience design in product life cycle management. In *2012 First International Workshop on Usability and Accessibility Focused Requirements Engineering (UsARE)*, pages 12–17, June 2012.

[2] David Baxter, James Gao, Keith Case, Jenny Harding, Bob Young, Sean Cochrane, and Shilpa Dani. A framework to integrate design knowledge reuse and requirements management in engineering design. *Robotics and Computer-Integrated Manufacturing*, 24(4):585–593, 2008. ICMR2005: Third International Conference on Manufacturing Research.

[3] Abdelhadi Belfadel, Jannik Laval, Chantal Bonner Cherifi, and Nejib Moalla. Requirements engineering and enterprise architecture-based software discovery and reuse. *Innovations in Systems and Software Engineering*, pages 1–22, 2022.

[4] Rodrigo Goncalves de Branco, Maria Istela Cagnin, and Debora Maria Barroso Paiva. Acctrace: Accessibility in phases of requirements engineering, design, and coding software. In *2014 14th International Conference on Computational Science and Its Applications*, pages 225–228, June 2014.

[5] Marjo Kauppinen. *Introducing requirements engineering into product development : towards systematic user requirements definition*. Doctoral thesis, 2005.

[6] Brijesh Kumar and HOD Dr Dhirendra Pandey. Requirements engineering process model add-on for software development. 2022.

[7] Ze-Lin Liu, Zhinan Zhang, and Yong Chen. A scenario-based approach for requirements management in engineering design. *Concurrent Engineering*, 20(2):99–109, 2012.

[8] Marian Mistler, Nadine Schlueter, and Manuel Löwer. Analysis of software tools for model-based generic systems engineering for organizations based on e-decode. In *2021 IEEE International Systems Conference (SysCon)*, pages 1–8, April 2021.

[9] Jorma Papinniemi, Lea Hannola, and Michael Maletz. Challenges in integrating requirements management with plm. *International Journal of Production Research*, 52(15):4412–4423, 2014.

[10] Thilo O. Richter, André Felber, Peter M. Troester, Albert Albers, and Kamran Behdinan. Visualization of requirements engineering data to analyse the current product maturity in the early phase of product development. *Procedia CIRP*, 91:271–277, 2020. Enhancing design through the 4th Industrial Revolution Thinking.

[11] Ahti Salo and Timo K. Kakola. Groupware support for requirements management in new product development. *Journal of Organizational Computing and Electronic Commerce*, 15(4):253–284, 2005.

[12] Emily Windisch, Constantin Mandel, Simon Rapp, Nikola Bursac, and Albert Albers. Approach for model-based requirements engineering for the planning of engineering generations in the agile development of mechatronic systems. *Procedia CIRP*, 109:550–555, 2022. 32nd CIRP Design Conference (CIRP Design 2022) - Design in a changing world.