GITHUB FOR PRODUCT DEVELOPMENT – HOW COULD THAT LOOK LIKE?

Hackenberg, Georg;
Zehetner, Christian;
Frühwirth, Dominik

School of Engineering, University of Applied Sciences Upper Austria

ABSTRACT

Product development is facing new challenges due to increasingly complex and individualized products in small batch sizes and short time to markets at high quality standards. Integrated product data management along with systematic requirements engineering and early stakeholder involvement are known to be key enablers for the success of future product development. In software development, established platforms such as GitHub exist, which have been shown to improve stakeholder communication, requirements elicitation, and software design decisions. In product development, similar platforms exist with impressive functionality, but which have some drawbacks such as closed source licenses, vendor-specific data formats, and expert-level user interfaces. To overcome the current situation, we study how the ideas of GitHub can be translated to an open source solution for product development and which concepts can be reused or must be changed. Core deliverables of our work are (1) an integrated data model of requirements (or design tasks), project schedules, and revisions of computer-aided design (CAD) models as well as (2) an interface model.

Keywords: Agile methodology, Project management, Requirements, Computer Aided Design (CAD), Version control

Contact:
Hackenberg, Georg
University of Applied Sciences Upper Austria
Austria
georg.hackenberg@fh-wels.at

Cite this article: Hackenberg, G., Zehetner, C., Frühwirth, D. (2023) 'GitHub for Product Development – How Could That Look Like?', in *Proceedings of the International Conference on Engineering Design (ICED23),* Bordeaux, France, 24-28 July 2023. DOI:10.1017/pds.2023.206

1 INTRODUCTION

Many researchers and practitioners including Salo and Kakola (2005) already noted that companies face increasing pressure to develop products with ever better functionality, higher quality, cheaper price, and shorter time-to-market. At the same time, their engineers need to deal with (partially) vague requirements specifications and their customers expect flexibility with respect to change requests – all due to uncertainties that cannot be resolved at the beginning of product development. Anitha and Prabhu (2012) showed that systematic requirements engineering with early stakeholder involvement and dedicated tool support represents half of the success of future product development.

For software development, today established platforms exist – such as GitHub[1] – which enable early stakeholder involvement based on an integrated view on requirements, schedules, and deliverables. Tsay et al. (2014) provided evidence that such platforms actually foster discussion among internal and external stakeholders, help to elicit software requirements for upcoming software releases, and make more appropriate software design decisions. Similarly, Lee et al. (2016) found that within the context of open source software development such platforms support the open innovation paradigm due to encouraging decentralized decision-making. Unfortunately, due to the nature of the deliverables (mainly source code, binaries, and documentation), these platforms cannot be applied to product development directly.

As Tony Liu and William Xu (2001) explained, in product development – traditionally – product data management (PDM) systems are used for storing design- and production-related product information as well as managing design and release workflows. Later, Grieves (2005) introduced the more holistic product lifecycle management (PLM) approach, which not only covers design- and production-related information, but covers everything from initial requirements analysis and planning to product disposal and recycling. Still, Houshmand and Valilai (2010) argued that due to the independence of the various software tools being used, isolated artifacts such as project plans, requirements specifications, and computer-aided design (CAD) models are generated, which lack a properly integrated and consistent data model. Later, Papinniemi et al. (2014) discussed that commercial product lifecycle management systems did not include actual requirement management capabilities and the integration of requirement analysis with product design information remained a challenge. More recently, Barth (2013) introduced platforms such as Dassault Systèmes 3DEXPERIENCE, which provide advanced integration of project management, requirements management, and product design activities based on an integrated vendor-specific data format, expert-level user interface, and closed source software implementation. Nevertheless, Marion and Fixson (2021) show that – today – many companies still reply on "classical" tool infrastructures comprising, e.g., Microsoft Project for project management, Microsoft Excel for requirements management, Dassault Systèmes Solidworks for product design, and email for team communication, while accepting tool barriers and their negative consequences.

1.1 Research goal

To overcome the current situation, we aim at developing an *open source* platform for product development based on *standard data formats* with primary focus adaptability and *ease-of-use*. We believe that both companies and the researchers will benefit from an open source approach because researchers can develop platform innovations faster and companies can integrate the platform easier into their infrastructures. Furthermore, we think that the platform should be as easy-to-use as GitHub to reduce the barrier for non-expert stakeholders to participate in the product development process, even if this requires omitting functionalities. Finally, we are convinced that standard data formats are the best (i.e. most efficient and most effective) way to interface with existing tool infrastructures at today's product development companies. From these goals we derive the following research question.

1.2 Research question

How can we translate the ideas from established, effective, and easy-to-use software development platforms such as GitHub to the domain of product development? Which elements of these platforms can we reuse and which elements do we need to adapt to support product development activities directly? To answer these questions, we deployed the following research methodology.

[1] https://www.github.com/

https://doi.org/10.1017/pds.2023.206 Published online by Cambridge University Press

1.3 Research methodology

We first reviewed the state-of-the-art on project management, requirements management and data integration in the context of product development and computer-aided design (see Section 2). From the results of the review as well as from informal expert discussions with Upper Austrian product development companies we derived a goal formulation and an initial set of requirements for the open source platform (see Section 3). Based on the initial set of requirements we started developing the platform in an iterative and incremental process including several informal feedback rounds with the previous experts over the course of 1.5 years with an approximate budget of 0.5 FTE (see Section 4).
One major result highlighted in this article is an integrated **data model** of requirements, schedules, and deliverables, which supports management of CAD model revisions, linking of requirements to parts and assemblies of CAD model revisions, asynchronous discussion about and clarification of requirements, as well as prioritization and scheduling of requirements through milestones with fixed start and end dates (see Section 4.1). A second major result highlighted here is an **interface model**, which we derived from GitHub, but which required substantial changes to support the underlying data model, corresponding operations, as well as individual stakeholder perspectives properly (see Section 4.2).
Finally, for the various feedback rounds we prepared a minimal case study to demonstrate the capabilities of the platform based on a 3D model of a LEGO buggy, which is provided free-of-charge by the Khronos Group[2]. Note that while the informal feedback we got from the experts is quite promising, a thorough evaluation of the platform efficiency, effectiveness and usability remains an open issue, which we intend to address with user studies in future work.

2 RELATED WORK

Subsequently, we provide an overview of the state-of-the-art on integrated project management, requirements management, and product design both from an **academic viewpoint** in Section 2.1 and from an **industrial viewpoint** in Section 2.2.

2.1 Academic viewpoint

The academic viewpoint comprises, besides others, **industry studies** showing the importance of data integration (see Section 2.1.1), **abstract frameworks** for data integration (see 2.1.2), as well as **concrete tools** for data integration in **product development** (see Section 2.1.3) and **concrete tools** for data integration in **other domains** (see Section 2.1.4).

2.1.1 Industry studies

Jassawalla and Sashittal (1998) study the importance of interdisciplinary collaboration in product development including customers and suppliers. Furthermore, the authors propose a collaboration framework based on principles such as shared responsibility as well as synergy between disciplines. Also, the authors stress success factors such as trust between project partners and support from top management.
Sanchez and Pérez (2003) also show the importance of collaboration in interdisciplinary product development teams. Therefore, the authors study practices in the automotive industry, while distinguishing between size of company and complexity of product. A main outcome is that strong collaboration results in better time-to-market and lower cost independent of company size and product complexity.
BüyüKözkan et al. (2004) highlight the importance of concurrent engineering and data integration in the agile manufacturing era. The authors summarize existing approaches, while distinguishing between collaboration, modeling and analysis, design synthesis and optimization, knowledge-based tools. Finally, they conclude that the tools must be improved with respect to integration and agility.
Durmuşoğlu and Barczak (2011) provide empirical evidence that software tools improve the effectiveness of product development projects. They use three measures to assess effectiveness of these projects: innovativeness, product quality, and market performance. The study suggests that the support for agile practices such as early stakeholder involvement should be improved in today's landscape.
Marion and Fixson (2019) have a deeper look into the tool landscape used during product development. The authors find that in general software tools help to share information and knowledge across team

[2] https://github.com/KhronosGroup/glTF-Sample-Models/tree/master/2.0/Buggy

https://doi.org/10.1017/pds.2023.206 Published online by Cambridge University Press

members, which applies specifically to virtual teams. However, the authors also suggest that general purpose tools might be easier to adopt by project teams than specialized solutions.

2.1.2 Abstract frameworks

Kauppinen (2005) proposes a framework for integrating requirements engineering effectively into organizations. Besides a high-level process model the framework considers the cultural change as a major factor in improving organizational practices. While the author shows how requirements engineering practices can be improved effectively, the author does not deliver concrete data models and tools.
Baxter et al. (2008) stress the importance of reusing knowledge generated in product development projects. Furthermore, the authors propose a framework for integrating requirements engineering with engineering design based on process, task, and product knowledge. However, the framework remains rather high-level and still does not provide a concrete technical realization of their ideas.
Papinniemi et al. (2014) propose a framework for better integrating requirements management into product lifecycle management tools. Their approach discusses the relation between customer requirements and product designs as well as the challenges for adapting existing practices in industry. However, the authors do not explain how their ideas can be implemented practically into existing tool landscapes.
Richter et al. (2020) propose a framework for visualizing the state and progress of product development projects. They summarize requirements of different stakeholders and propose a library of visualization techniques for different aspects of project progress. However, the authors do not provide ideas on how to store requirements, link requirements to product designs, and derive process information practically.

2.1.3 Concrete tools for product development

Salo and Kakola (2005) evaluate the effectiveness of online collaboration platforms for requirements management in product development. Furthermore, the authors summarize the requirements for such platforms and propose a design, which is tested across several large-scale projects. However, the authors do not consider the integration of requirements with project schedules and design revisions.
Liu et al. (2012) provide a scenario based approach for the elicitation, decomposition, and formalization of engineering design requirements. Their approach tries to increase the quality of requirements specifications in the context of product development. While their approach provides better guidance on how to express requirements, their approach still lacks integration with project schedules and design revisions.
Windisch et al. (2022) propose a model-based approach for expressing requirements in product development. Their approach is based on an activity-based view of design and verification tasks, which need to be carried out by product engineers. Through integration into agile project management tools, the requirements can be linked to project schedules, but the link to product design revisions is missing.

2.1.4 Concrete tools for other domains

Goncalves de Branco et al. (2014) propose a computer-aided software engineering (CASE) tool which helps to build accessible products. Their approach is based on an ontology, that describes technical implementations of accessibility requirements and, hence, enables traceability. Their approach is well suited for the software domain and accessibility, but cannot be used directly in product development.
Belfadel et al. (2022) propose a framework and tool for matching customer requirements with existing enterprise capabilities. Their approach is based on a multi-layer model of the enterprise architecture as well as patterns of customer requirements. The tool supports discovery of feasible capabilities and effectively fosters reuse, but its application to agile product development remains unclear.
Mistler et al. (2021) propose a modeling technique and select appropriate tooling for matching business requirements with organizational structures. To select the tooling, the authors collect requirements that must be fulfilled in order for the tooling to be applicable. Again, the approach helps in arranging and combining enterprise capabilities, but its application to product development remains unclear.

2.2 Industrial viewpoint

The industrial viewpoint comprises, besides others, the platforms of the major players **Dassault Systèmes 3DEXPERIENCE**, **Siemens Teamcenter**, and **PTC Windchill/Onshape**. These solutions all provide an integrated data model across the entire product lifecycle from requirements analysis and

https://doi.org/10.1017/pds.2023.206 Published online by Cambridge University Press

planning to product disposal and recycling. However, besides the great innovations that have been implemented we see three major issues: First, the solutions are closed source limiting platform innovations driven by the research community as well as adaptability to company-specific needs backed by a broader service provider market. Second, the data formats are vendor-specific and the available interfaces only expose a subset of information limiting advanced tool integrations and automations across vendors. And third, the user interfaces typically target expert-level users inhibiting the participation of non-expert stakeholders in the product development process due to a steep learning curve.

3 PLATFORM GOALS AND REQUIREMENTS

Before designing a new open source, easy-to-use platform for information exchange and collaboration between customers, project managers, requirements engineers, and product designer, we summarize our goals and derive concrete requirements. We use the requirements to verify our platform design, and we plan to use the goals to validate the effectiveness of the underlying approach.

3.1 Platform goals

Our main goal is to improve the information flow between customers, project managers, requirements engineers, and product designers, and to reduce inaccuracies and inconsistencies in product development. In particular, we want to make it easier for customers to participate in product development processes, to understand the current state of running projects, and to provide valuable feedback for validation. At the same time we want to make it easier for project managers, requirements engineers, and product designers to share and validate their individual view points and, hence, build a mutual understanding faster.

3.2 Platform requirements

In a first step, we translated the previous goals into functional requirements (FR) and non-functional requirements (NFR):

1. Customers as well as project managers, requirements engineers, and product designers shall be able to authenticate and shall have different permissions. (FR)
2. Project managers shall be able to start new projects and assign customers, requirements engineers and product designers to these projects. (FR)
3. Product designers shall be able to upload CAD models and maintain version histories independent of the CAD tool vendor used. (FR)
4. Customers, project managers, requirements engineers, and product designers shall be able to create and discuss design tasks to be completed. (FR)
5. It shall be possible to refer to parts and assemblies of CAD models during discussion of design tasks to improve the quality of information exchange. (FR)
6. Customers shall be able to define priorities and set deadlines for design task execution, as well as track progress of design task execution. (FR)
7. Project managers, requirements engineers, and product designers shall be able to report progress of design task execution. (FR)
8. It shall be possible to use any device from smartphone over tablet to laptop and desktop to access the features anywhere and anytime. (NFR)
9. It shall be possible to extend the user interface with support for virtual reality and augmented reality hardware in the future. (NFR)

In a second step, we derived a concrete platform design from the previous list of requirements.

4 PLATFORM DESIGN

As explained previously, our platform design comprises a **data model** and an **interface model**. The data model provides an integrated representation of stakeholders, roles, CAD model revisions, design tasks, and project schedules. In contrast, the interface model demonstrates how a graphical user interface could look like for the application. In the following, we expain both models in more detail.

https://doi.org/10.1017/pds.2023.206 Published online by Cambridge University Press

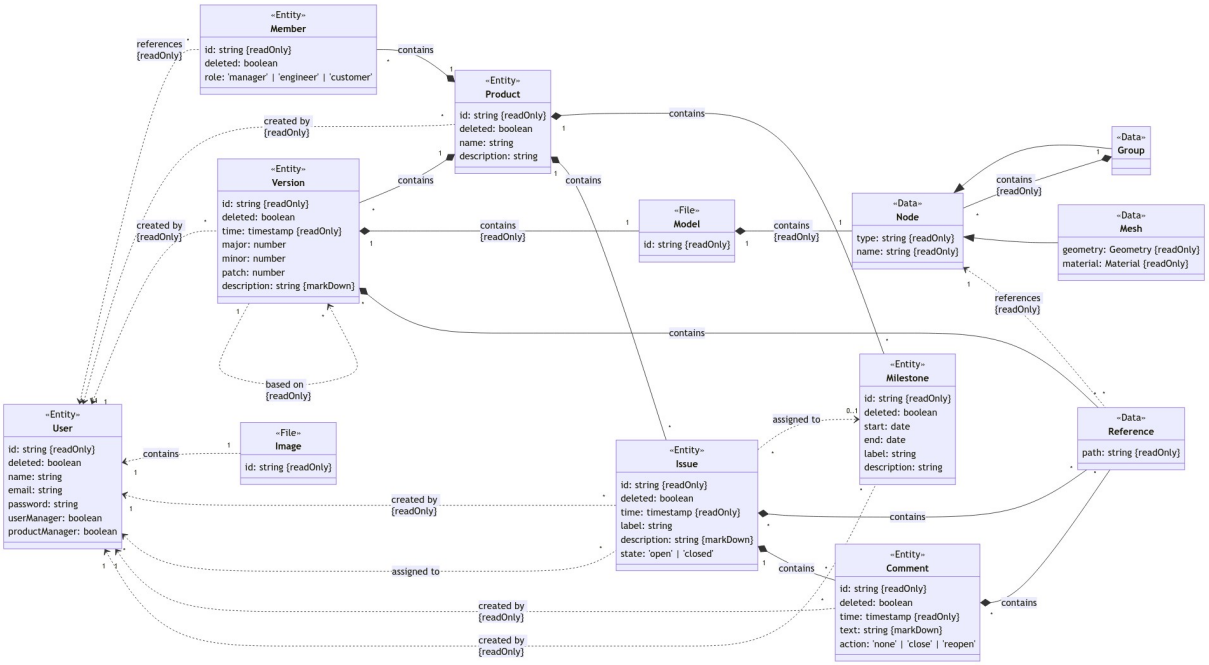


Figure 1. Integrated data model for improved information exchange between customers, project managers, requirements engineers, and product designers

4.1 Data model

This section describes the integrated data model as shown in Figure 1. The data model contains the following entities: *User*, *Product*, *Version*, *Issue*, *Comment*, *Milestone*, and *Member*. Note that all entities share a common identifier attribute, which is used for referencing and linking entities. Furthermore, all entities share a common deleted attribute, which is used to hide entities after removal.

The *User* entity represents the individual stakeholders, who access the tool during product development. The entity defines an email and a password attribute, which are used for authentication purposes. Additionally, the entity defines a name attribute, which is used for a human-readable identification of stakeholders. Similarly, the entity links an *Image* file, which provides a human-interpretable visual representation of stakeholders (i.e. a profile picture). Furthermore, the entity defines a user manager and a product manager flag, which are used for permission control as explained later.

The *Product* entity represents the individual products or product development projects managed with the tool. The entity is always linked to a *User* entity, representing the stakeholder who created the product in the first place. Then, the entity defines a name attribute, which is used to identify the product in a human-readable manner. Furthermore, the entity defines a description attribute, which is used to explain the purpose of the product only briefly.

The *Member* entity represents the permission to access certain product data through the tool. Consequently, the entity is linked to a *Product* entity, representing the product for which access is granted. Also, the entity is linked to a *User* entity, representing the stakeholder who is granted product access. Finally, the entity defines a *role* attribute, which controls the permission level as explained later, and which can have one of three values: *manager*, *engineer*, or *customer*.

The *Version* entity represents a revision of the product design created by a product designer. The entity is linked to a *Product* entity to identify the product, for which the design revision was created. Similarly, the entity is linked to a *User* entity to identify the product designer, who was responsible for creating the design revision. Furthermore, the entity is linked to previous *Version* entities to document the history of design revisions including version branching and merging. Then, the entity defines a major, a minor, and a patch number, which together represent a version number according to semantic versioning[3] practices. Also, the entity defined a description, which provides a human-readable explanation of the

[3] https://semver.org/

https://doi.org/10.1017/pds.2023.206 Published online by Cambridge University Press

design changes that have been applied. Finally, the entity links a *Model* file, which represents the actual design data including assembly and part structures as well as geometry and material information.
We work with a generic representation of *Model* files, which is independent of the CAD tool vendor and data format used. We assume a model contains *Node* objects, which carry all the engineering information created by the product designer. Furthermore, we assume a node defines a name attribute, which can be used as for human-readable identification of nodes. Additionally, we assume a node provides a type attribute, which can be used to distinguish different types of nodes. At the moment, we distinguish two types of nodes, namely *Group* nodes and *Mesh* nodes. Groups represent the assembly structures and provide links to child nodes being assembled, which can be both groups and meshes. Meshes, on the other hand, represent the atomic parts of the product design such as screws or rivets, and carry geometry and material information. Technically, we work with the glTF[4] format for representing models including their node structures as well as their geometry and material information. However, the STEP[5] format or the COLLADA[6] format could be used equally well, since they follow the same principles. In fact, even a vendor-specific data format could be used as long as appropriate parsers are available.
The *Milestone* entity essentially represents project schedules including deadlines for the realization of design tasks. The entity always links a *Product* entity to identify the product or project the milestone belongs to. Furthermore, the entity links a *User* entity to identify the product manager, who created the milestone. Finally, the entity defines a start and an end date, which represent the time frame for working on the milestone and achieving its goals.
The *Issue* entity represents design tasks, which have to be performed by product designers during product development projects. The entity is always linked to a *Product* entity to identify the product or project the issue belongs to. Furthermore, the entity is linked to multiple *User* entities to identify the one stakeholder, who reported the issue in the first place, and to identify the product designers, who are responsible for issue resolution. Moreover, the entity can be linked to a *Milestone* entity to identify the time frame, within which the issue should be resolved. Then, the entity defines a time attribute, which records the point in time when the issue was reported originally. Also, the entity defines a label attribute, which provides a human-readable summary of the issue. Additionally, the entity defines a text attribute, which provides a more detailed explanation of the design task including Markdown-based[7] *Reference* objects as explained later. Finally, the entity defines a state attribute, which enables us to distinguish between open and closed design tasks.
The *Comment* entity represents discussions between stakeholders on issues or design tasks respectively. The entity links an *Issue* entity to identify the issue or design task the comment belongs to. Furthermore, the entity links a *User* entity to identify the stakeholder, who posted the comment. Then, the entity defines a time attribute, which records the point in time when the comment was posted. Additionally, the entity defines a text attribute, which contains the actual content of the comment including Markdown-based *Reference* objects as explained later. Finally, the entity defines an action attribute, which can be used to close or reopen an issue by posting a comment.
As explained previously, we work with a Markdown-based representation of *Reference* objects, which can be contained in the description of *Issue* entities as well as *Comment* entities. These references can be used to refer to *Node* objects, that are contained in the CAD models of design revisions for a particular product or project. Consequently, the specification and discussion of design tasks can be enriched with links to assembly structures and parts, which have been designed and delivered previously.

4.2 Interface model

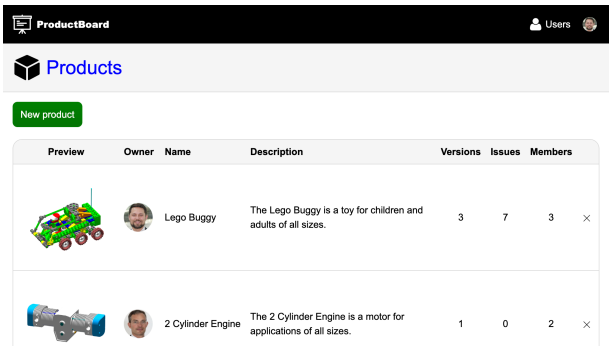
The interface model provides different views managing the individual entities and their relationships as introduced in the previous section. Here, due to space limitations we concentrate on the most important parts of the interface model: The *product overview*, the *version overview*, the *issue detail view*, and the *milestone detail view*. In the following, we explain each view in more detail.

---

[4] https://www.khronos.org/gltf/
[5] https://www.iso.org/standard/66654.html
[6] https://www.khronos.org/collada/
[7] https://www.Markdownguide.org/

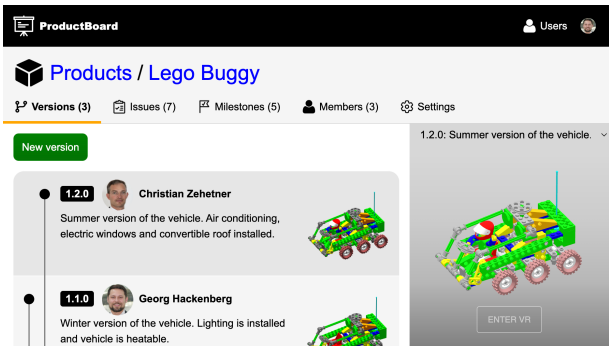
https://doi.org/10.1017/pds.2023.206 Published online by Cambridge University Press

4.2.1 Product overview



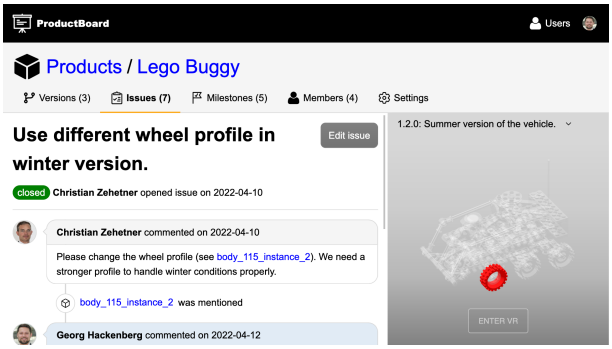
The *product overview* shows the products managed on the platform and visible to the user. On the top of the page, users with the product manager permission are provided a button to create new products. Then, for each existing product a preview is shown of the latest CAD model revision uploaded. Furthermore, the name of the product owner is shown, i.e. the user with the product manager permission who has created the product. Also, the human-readable name and description of the product are displayed to help the user navigate. Moreover, the number of CAD model revisions, the number of issues, and the number of members are listed for each product to provide an impression of user activity. Additionally, users with product manager permission can see a cross button for each product, which allows them to delete the product from the database and from the list. Finally, when clicking on the product, the user is redirected to the *version overview* as introduced in the next section.

4.2.2 Version overview



The *version overview* shows the CAD model revisions that have been uploaded for the selected product. On the top of the page, users with the engineer member role are provided a button to create a new CAD model revision. Then, for each CAD model revision the version number, the name of the user who created the version, a human-readable description of the changes that have been made, and links to respective base versions are displayed. Note that for now the user selects the base versions manually when creating new CAD model revisions. Furthermore, the user can select either none, one, or multiple base versions and, hence, describe *branch* and *merge* operations. Additionally, previews of the different CAD model revisions are provided to help users understand the version history. Finally, when clicking on a product version the associated CAD model revision is loaded into an interactive 3D view on the right side. In this view the CAD model can be zoomed, rotated, and panned using mouse interactions or a virtual reality headset.

4.2.3 Issue detail view

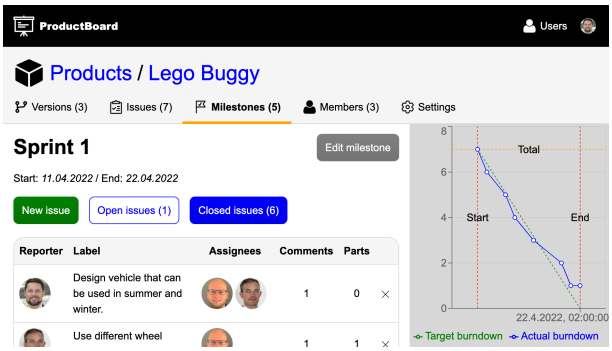


Then, the *issue detail view* shows the discussion between the stakeholders around a design task. On the top of the page the name of the issue is displayed, which must be defined when creating the design task. Below the issue state (open or closed), the name of the user who created the issue, and the creation date are shown. Afterwards, a list of comments is provided, which are attached to the issue. For each comment the profile picture and the name of the user, who created the comment, as well as the creation date are shown. Furthermore, for each comment a human-readable description is shown, which can include markdown-based links to parts and assemblies of defined CAD model revisions. Finally, on the right side a 3D view is provided, which allows one to select and display any CAD model revision of the product. In this 3D view parts and assemblies are

https://doi.org/10.1017/pds.2023.206 Published online by Cambridge University Press

highlighted, which are referenced in the comments. New markdown-based references can be inserted into a comment field at the bottom of the page by mouse click onto the desired part or assembly.

4.2.4 Milestone detail view



The *milestone detail view* shows the current sprint including open and closed design tasks. At the top of the page the name of the milestone as well as the start and end dates are shown. Then, for users with the manager member role a button is provided, through which new milestones can be created. Afterwards, the list of issues is displayed, which are assigned to the current milestone and which can be filtered by state (open or closed). Furthermore, the list shows for each issue the profile pictures of the reporting and the assigned users, the issue name, as well as the number of comments and referenced parts. Finally, on the right side a burn down chart illustrates the progress of the current milestone. The horizontal axis of the chart shows the duration of the milestone from start to end date. The vertical axis of the chart shows the number of open issues instead. Then, a target burn down line illustrates the ideal speed of issue resolution, while the actual burn down line shows the real speed. This illustration helps the product manager to track progress.

5 CONCLUSION

In this article, we summarized our progress on translating the ideas of GitHub to the domain of product development. Therefore, we first developed an integrated data model for requirement (or design tasks), project schedules, and CAD model revisions. Most importantly, we realized the integration between requirements and CAD model revisions with Markdown syntax, which proved suitable. Furthermore, we developed an interface model, through which product versions can be uploaded as well as issues and milestones can be managed. The interface model is tightly aligned with GitHub, but we made substantial changes in the revision graph and the issue view to incorporate interactive 3D models. In the next step, we plan – besides others – to conduct user tests and extend the requirement management capabilities.

REFERENCES

Anitha, P. and Prabhu, B. (2012), "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)*, pp. 12–17.

Barth, A. (2013), "3d experiences – dassault systèmes 3ds strategy to support new processes in product development and early customer involvement", in: G.L. Kovács and D. Kochan (Editors), *Digital Product and Process Development Systems*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 24–30.

Baxter, D., Gao, J., Case, K., Harding, J., Young, B., Cochrane, S. and Dani, S. (2008), "A framework to integrate design knowledge reuse and requirements management in engineering design", *Robotics and Computer-Integrated Manufacturing*, Vol. 24 No. 4, pp. 585–593. ICMR2005: Third International Conference on Manufacturing Research.

Belfadel, A., Laval, J., Bonner Cherifi, C. and Moalla, N. (2022), "Requirements engineering and enterprise architecture-based software discovery and reuse", *Innovations in Systems and Software Engineering*, pp. 1–22.

Goncalves de Branco, R., Cagnin, M.I. and Barroso Paiva, D.M. (2014), "Acctrace: Accessibility in phases of requirements engineering, design, and coding software", in: *2014 14th International Conference on Computational Science and Its Applications*, pp. 225–228.

BüyüKözkan, G., Derelİ, T. and Baykasoğlu, A. (2004), "A survey on the methods and tools of concurrent new product development and agile manufacturing", *Journal of Intelligent Manufacturing*, Vol. 15 No. 6, pp. 731–751.

Durmuşoğlu, S.S. and Barczak, G. (2011), "The use of information technology tools in new product development phases: Analysis of effects on new product innovativeness, quality, and market performance", *Industrial Marketing Management*, Vol. 40 No. 2, pp. 321–330. Special issue on Service-Dominant Logic in Business Markets.

https://doi.org/10.1017/pds.2023.206 Published online by Cambridge University Press

Grieves, M.W. (2005), "Product lifecycle management: the new paradigm for enterprises", *International Journal of Product Development*, Vol. 2 No. 1-2, pp. 71–84.
Houshmand, M. and Valilai, O.F. (2010), "Collaborative information system architecture for cad/cam in new product development based on step standard", in: *Proceedings of the World Congress on Engineering and Computer Science*, Vol. 2, pp. 1072–1080.
Jassawalla, A.R. and Sashittal, H.C. (1998), "An examination of collaboration in high-technology new product development processes", *Journal of Product Innovation Management*, Vol. 15 No. 3, pp. 237–254.
Kauppinen, M. (2005), *Introducing requirements engineering into product development : towards systematic user requirements definition*, Doctoral thesis, Helsinki University of Technology.
Lee, J., Min, J. and Lee, H. (2016), "The effect of organizational structure on open innovation: A quadratic equation", *Procedia Computer Science*, Vol. 91, pp. 492–501. Promoting Business Analytics and Quantitative Management of Technology: 4th International Conference on Information Technology and Quantitative Management (ITQM 2016).
Liu, Z.L., Zhang, Z. and Chen, Y. (2012), "A scenario-based approach for requirements management in engineering design", *Concurrent Engineering*, Vol. 20 No. 2, pp. 99–109.
Marion, T. and Fixson, S. (2019), "The influence of collaborative information technology tool usage on npd", *Proceedings of the Design Society: International Conference on Engineering Design*, Vol. 1 No. 1, p. 219–228.
Marion, T.J. and Fixson, S.K. (2021), "The transformation of the innovation process: How digital tools are changing work, collaboration, and organizations in new product development*", *Journal of Product Innovation Management*, Vol. 38 No. 1, pp. 192–215.
Mistler, M., Schlueter, N. and Löwer, M. (2021), "Analysis of software tools for model-based generic systems engineering for organizations based on e-decode", in: *2021 IEEE International Systems Conference (SysCon)*, pp. 1–8.
Papinniemi, J., Hannola, L. and Maletz, M. (2014), "Challenges in integrating requirements management with plm", *International Journal of Production Research*, Vol. 52 No. 15, pp. 4412–4423.
Richter, T.O., Felber, A., Troester, P.M., Albers, A. and Behdinan, K. (2020), "Visualization of requirements engineering data to analyse the current product maturity in the early phase of product development", *Procedia CIRP*, Vol. 91, pp. 271–277. Enhancing design through the 4th Industrial Revolution Thinking.
Salo, A. and Kakola, T.K. (2005), "Groupware support for requirements management in new product development", *Journal of Organizational Computing and Electronic Commerce*, Vol. 15 No. 4, pp. 253–284.
Sanchez, A. and Pérez, M.P. (2003), "Flexibility in new product development: a survey of practices and its relationship with the product's technological complexity", *Technovation*, Vol. 23 No. 2, pp. 139–145.
Tony Liu, D. and William Xu, X. (2001), "A review of web-based product data management systems", *Computers in Industry*, Vol. 44 No. 3, pp. 251–262.
Tsay, J., Dabbish, L. and Herbsleb, J. (2014), "Let's talk about it: Evaluating contributions through discussion in github", in: *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2014, Association for Computing Machinery, New York, NY, USA, p. 144–154.
Windisch, E., Mandel, C., Rapp, S., Bursac, N. and Albers, A. (2022), "Approach for model-based requirements engineering for the planning of engineering generations in the agile development of mechatronic systems", *Procedia CIRP*, Vol. 109, pp. 550–555. 32nd CIRP Design Conference (CIRP Design 2022) - Design in a changing world.

https://doi.org/10.1017/pds.2023.206 Published online by Cambridge University Press