TReqs: Tooling to enable teams to manage system

Yu-Jhen Chen*

Supervisor: Eric Knauss

* yujhen@student.chalmers.se

I. Introduction

A. Background

As mentioned by De Lucia et al. [2], a requirements management tool should enable traceability of requirements. Therefore, robust IDs management is essential for both managers and development teams to communicate and keep track of requirements. The reason for using this T-reqs is because an agile team works in parallel, there is a need of enabling team members to generate requirements by themselves. Also, because the team wants to use Git to develop, there's no requirements management tool that supports Git.

B. Aim

The initial version of T-reqs was proposed by Knauss [5]. Gebremichael developed a requirements management tool (T-reqs 2.0) [3] based on Knauss's version. The study aims at finding a viable and simple solution for requirements IDs management of requirements management tool to fulfill the needs of agile developing teams. Besides, a requirements management tool (T-reqs 3.0) is developed according to Gebremichael's T-reqs 2.0. T-reqs 3.0 is to realize the requirements IDs management solution and implement new workflows of T-reqs' s functions. The study also compares various solutions for requirements IDs management and analyzed the pros and cons of these solutions. Furthermore, the workflows of functions in T-reqs 3.0 are also compared with those in T-reqs 2.0.

II. Related Work

According to De Lucia et al. [2], there are five requirements engineering activities. Among these five activities: they proposed three methods for Requirements Documentation, which are: realize documentation and development within several team members, digitalize documentation, apply reverse engineering process to produce documentation; while, Requirements Validation contains main activities which are Requirements reviews, Unit testing, Evolutionary prototyping, and Acceptance testing; Finally, Requirements Management focus on a requirements traceability.

Kasauli et al. [4] answered three questions from the perspective of requirements engineering through multiple-case study. Based on two questions among them, they addressed different challenges and categorized them into six types. Furthermore, they proposed a framework that included these challenges and corresponding solutions.

Knauss et al. [6] did the research based on the product development team of a company, which changed into agile development processes currently. Because development processes are changed, requirements management is changed. These changes lead the existing requirements tool to be no longer to fulfill the needs of requirements management especially two requirements of requirements tool. Therefore, Knauss et al. [6] focus on studies of requirements management with tooling that meet the needs of the company, and concluded functions that T-reqs need according to user stories and existing challenges of requirements tooling.

Yazan and Gabriel-Marian [1] addressed common challenges and proposed solutions of an innovative requirements tool (T-reqs) by bringing this tool to a software requirements engineering second-cycle course. According to their findings, T-reqs has several problems such as the complexity of environment setting when using T-reqs; the compatibility T-reqs for different systems; manual manner; the necessity of having nature languages knowledge and requirements knowledge. Furthermore, based on these problems, authors proposed some features that should be included into the future T-reqs: have friendly UI; should include RE learning tool in T-reqs; should modeling features of T-reqs that allows developers to write requirements and trace requirements; require test and validation of a product at the final development stage.

Gebremichael and Mebrahtom [3] proposed an upgraded version of T-reqs based on

previous features of T-reqs. The new T-reqs includes features: Configuration and template files generation, IDs generation and reporting generated IDs history, Model support for requirements specification, Multi-repository support, Change history of requirements, and Validation report. The new T-reqs has two work-flows: first of all, a database setting of Amazon Relational Database Service (Amazon RDS) is required to manage IDs; secondly, the environment setting on the local side is required when using new T-reqs. Furthermore, the authors addressed large-scale agile RE challenges and proposed solutions that are provided by new T-reqs.

III. T-REQS TOOLING

T-reqs that proposed by Gebremichael and Mebrahtom [3] has five main functions: Initialize T-reqs, Generate IDs, Check requirements consistency, Get generated IDs, and Generate model previews. Figure 1 shows the workflows of each function. To execute these five functions, developers should enter corresponding commands in the command line. Table I shows each command and corresponding function of T-reqs.

TABLE I

T-REQS COMMANDS AND FUNCTIONS

command	**functions**
treqsconfig	Initialize T-reqs
generateid	Generate IDs
treqs	Check requirements consistency
generatedid	Get generated IDs
generatepreview	Generate model previews

1) Initialize T-reqs: To execute T-reqs on a local development environment, developers should first install python and create a database and related tables (see [3]) in Amazon Relational Database Service (Amazon RDS). After finishing all the environment settings, developer input *treqsconfig* command in a command line, and input necessary information (such as a directory of the local project) into the command line. T-reqs writes this information into a .json file to store parameters and configurations of the local development environment. After that, T-reqs generates templates of requirements documents under its own repository. Developers can either use existing requirements documents in the local

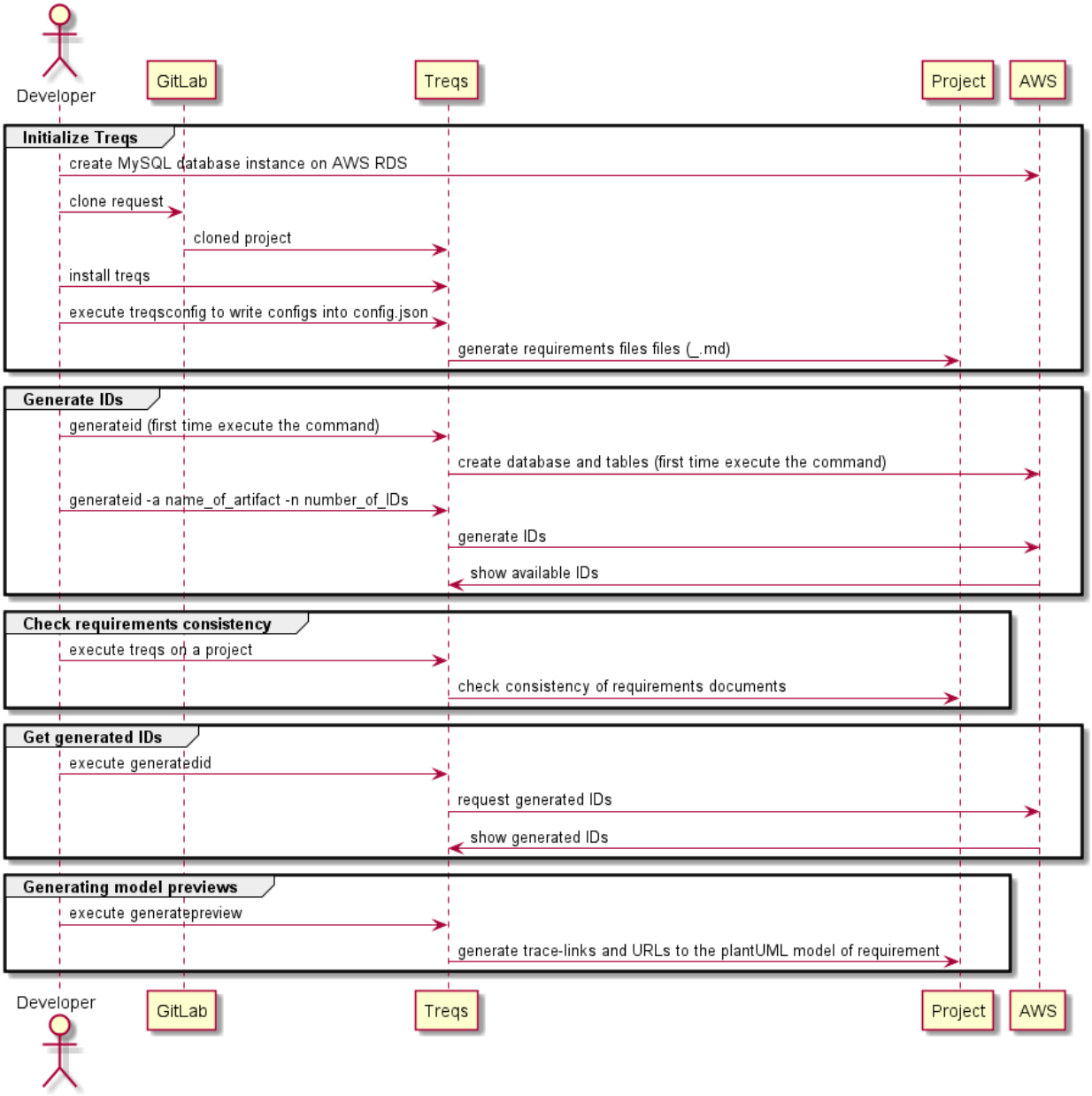


Fig. 1. work-flow of T-reqs from [3]

development environment or use those generated templates to maintain and manage requirements documents.

2) Generate IDs: When developers want to create new requirements and do programming in the local development project for new requirements, developers should enter *generateid* command in a command line. When the command *generateid* is executed, T-reqs checks if it is the first time that developers want to create a new requirement of the project. If the

developer has not generated any requirement before, T-reqs creates databases and necessary tables (such as user stories, system requirements, quality requirements, test cases, and accept requirements) in the Amazon database. After all necessary tables are created, T-reqs requires developers to input necessary fields (such as the type of requirement) and inserts an ID and relevant information of the requirement in tables of the Amazon database.

3) Check requirements consistency: Before developers commit local changes to a version control system, developers should make sure all the changes are correct by checking requirements consistency. When the command *treqs* is executed, T-reqs checks all the consistency of requirements documents and writes summary automatically into log files. Once T-reqs finishes consistency checking, developers should look into the log files and fix problems of requirements documents. When T-reqs prompts out a message that informs developers of passing all consistency, developers can commit the local changes to the version control system.

4) Get generated IDs: When a command *generatedid* is executed, T-reqs accesses to the Amazon RDS account of the developer and obtains all IDs generated by the developer. This function allows developers to check the latest IDs of requirements and other information such as ID generated date.

5) Generate model previews: Developers can execute the command *generatepreview* to view plantUML models that are from plantUML code. Each plantUML model is linked to a requirement in the corresponding requirements document.

Since developers need to either have an Amazon RDS account or use an Amazon RDS account that owns by their team when developing, members of a team have to spend extra efforts to maintaining the Amazon RDS. Besides, two requirements may be linked to the same function of a system, in which case the requirements need to be merged. After merging two requirements into one requirement, the old requirement IDs in the same requirement document become incorrect. In addition, since a requirement in a requirements document may not be accepted during the review step, the ID that links to a requirement is no longer used. Unused IDs in Amazon RDS will cause incorrect IDs when developers generate new requirements next time.

In summary, using Amazon RDS to manage IDs may lead to incorrect requirements IDs. Besides, because the development team has to manage Amazon accounts, the adoption of T-req in development projects becomes complicated. This brings a negative effect on the

use of T-req, which is not convenient for team members.

IV. Requirements IDs Management

In this section, in addition to the Amazon RDS requirements IDs management, there are two solutions for requirements IDs management are introduced. These two solutions aim at solving the issues in Amazon RDS requirements IDs management.

A. Amazon RDS

The Architecture of [3] IDs management solution that uses Amazon RDS is as shown in Figure 2. This solution focuses on providing databases of Amazon RDS for teams to manage IDs. In this solution, each team has its own databases to manage requirement IDs. Each requirement in the requirements document is linked to the corresponding requirement ID in the database.

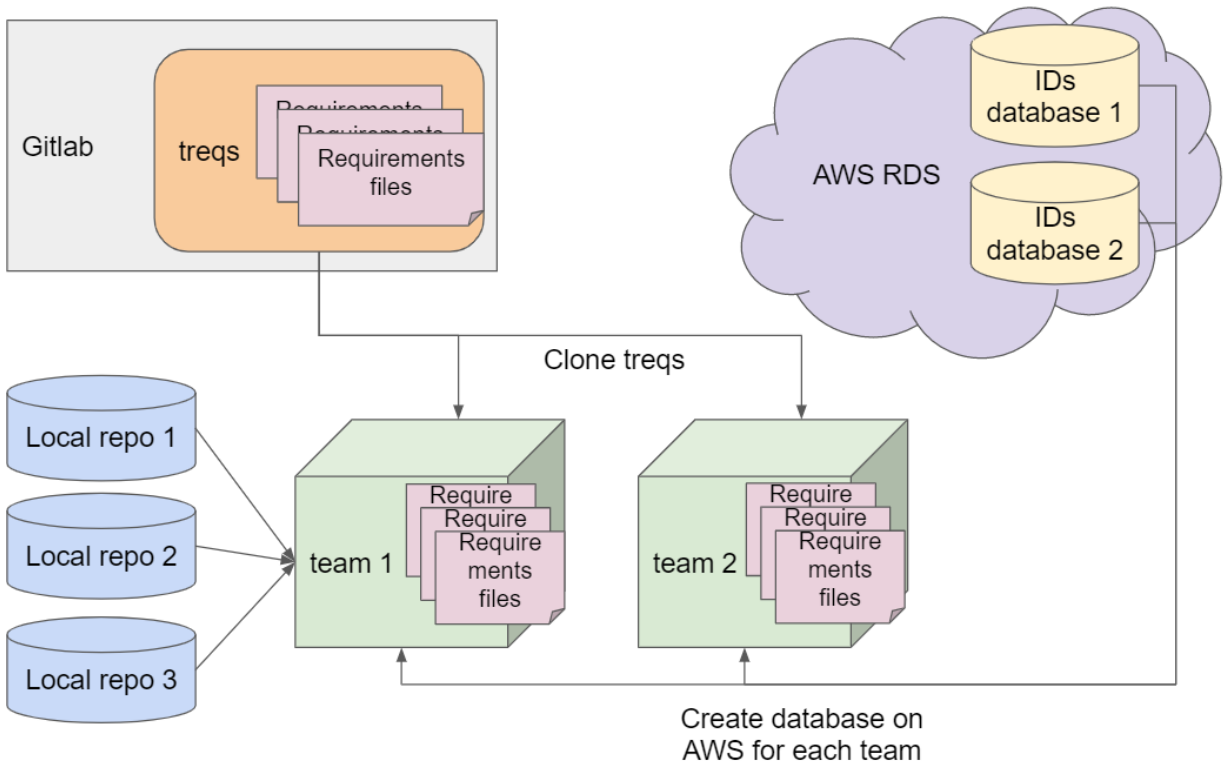


Fig. 2. Architecture of IDs management solution from [3]

B. Unique ID

Figure 3 is an proposed architecture for requirements IDs management, which utilizing the idea of unique IDs. Different from the solution of using Amazon RDS [3], this solution enables a team to only manage their requirements documents rather than depending on third-party service (which is Amazon RDS).

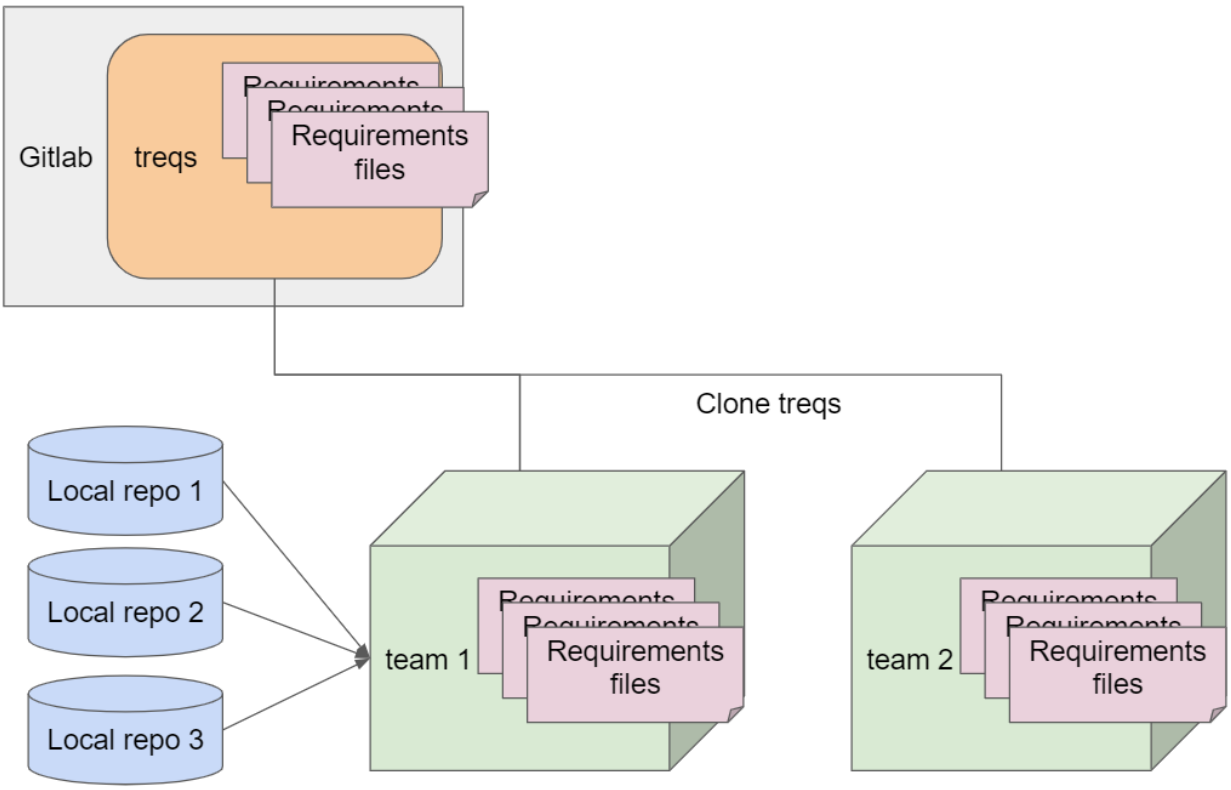
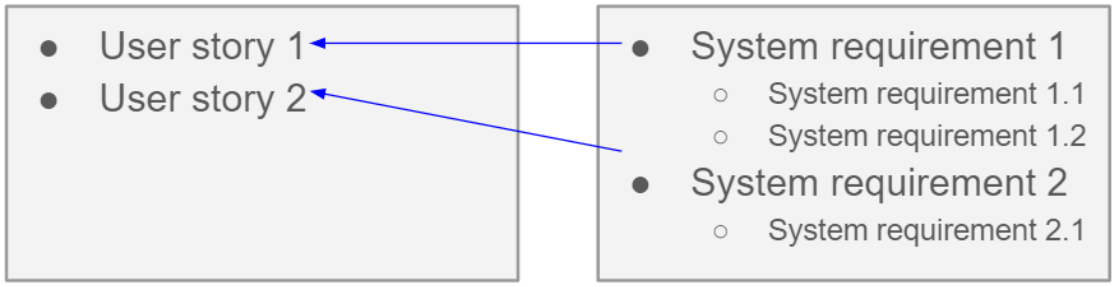


Fig. 3. Architecture of new IDs management solution (Requirements IDs are generated and stored in requirements documents rather third-party which is Amazon RDS)

Figure 4 provides an example of how unique IDs enable traceability of user stories and system requirements. The picture on the top in Figure 4 shows that in requirements documents, a system requirement links to a user story (system requirement 1 links to user story 1). To achieve traceability (so that system requirements can be linked to user stories), a unique id (uid) for each requirement is stored. The table in the middle of Figure 4 shows basic fields for storing necessary information of a requirement. The example here is user stories: uid is a unique id of a requirement (user story); customID is used for communication between managers and the development team. A customID is formatted

by its requirement type (US means user story) and a number (which is the order of this requirement in a requirement document); the prefix is the requirement type. The second table of Figure 4 shows basic fields that store information of system requirements. Among these fields, linked_us is the corresponding uid of user stories; parent_sr indicates the current requirement is a child requirement of which requirement.



uid	customID	prefix
14dfb9d4	US1	US
14dfbbf0	US2	US

uid	customID	prefix	linked_us	parent_sr
b47100f8	SR1	SR	14dfb9d4	root
b4710314	SR1.1	SR	14dfb9d4	b47100f8
b4710404	SR1.2	SR	14dfb9d4	b47100f8

Fig. 4. An example of tracing uid to user story and system requirements

There are two method of generate a new requirement: generate a requirement with customID, and generate a requirement without customID.

1) Generate requirements with customID: This method is as shown in Figure 5. When a developer generates a requirement, a customID is generated. The value of this customID is based on the order of where this new requirement is put in the requirements document. However, because customID is an ID that follows a certain order in a requirements document, the already existing customIDs in the same requirements document have to be updated if the new generated requirement is inserted in the middle of the requirements document.

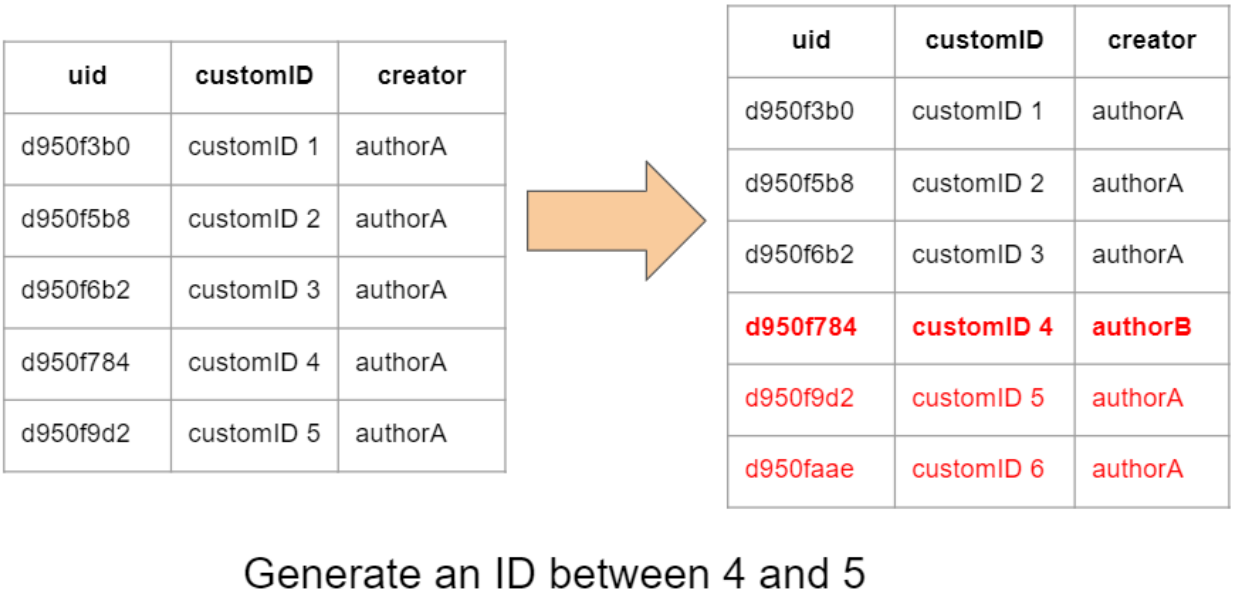


Fig. 5. Generate requirements with customID

2) Generate requirements without customID: Figure 5 shows the method of generating requirements without customID. In this method, When a developer generates a new requirement, only the necessary fields of the requirement are generated, such as uid and the information of the developer. CustomIDs are generated when managers or developers want to see the preview of a requirements document, which means these customIDs are not stored in requirements documents. Therefore, what a developer needs to do is maintain the newly generated requirement, he or she does not need to update the existing requirements in the requirements document.

The method of generating requirements without customID is chosen as the proposed IDs management solution. The main reason is that developers should take their own responsibilities of development, which means that any change of the documents or source code should be traceable to the developers. The method of generating requirements with customID leads to a problem that updates of customIDs may be done by other developers instead of the developer who created the requirements originally.

V. New version of T-reqs Tooling

A new version of T-reqs Tooling is developed based on the T-reqs Tooling project of [3]. As mentioned above, the method of generating requirements without customID is adopted,

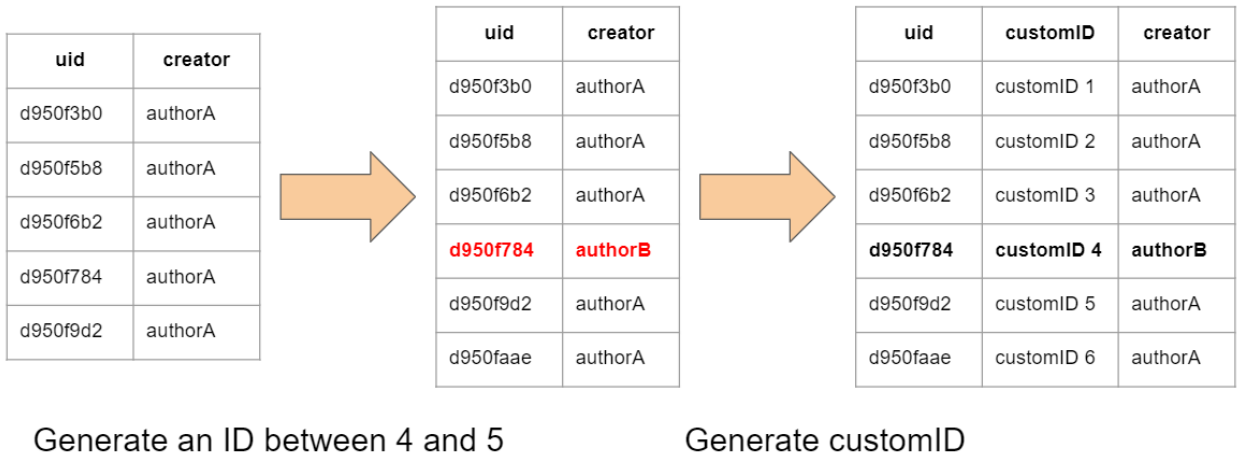


Fig. 6. Generate requirements without customID

which use unique IDs to realize requirements IDs management.

A. Functions and workflows

The new T-reqs has three functions: Generate requirement, Check consistency, and Generate requirements document. The following is an introduction to each function.

1) Generate requirement: Generate requirement is a function for developers when developers need to create new requirement into requirements documents. Figure 7 is the workflow of Generating system requirement. The function Generate requirement is triggered by executing a command *generatereq*. When Generate requirement is executed, the basic information that T-reqs requires includes requirement type, the path of requirement document, and parent requirement in the document (if any). As shown in Figure 7, since a system requirement links to a user story, developers should choose linked requirement (which is a user story) when generating system requirements. After developers provide all required fields, T-reqs generates a requirement tag based on developers' inputs. The developer should copy the output to the target requirements document.

2) Generate requirements document: The function Generate requirements document is used to generate a requirements document which includes numerical customIDs. After the developer executes a command *generatereqdoc*, T-reqs requires the developer to input requirement type and the path of the target requirements document. Later, T-reqs sorts all requirements in the target requirements document and generates corresponding customID

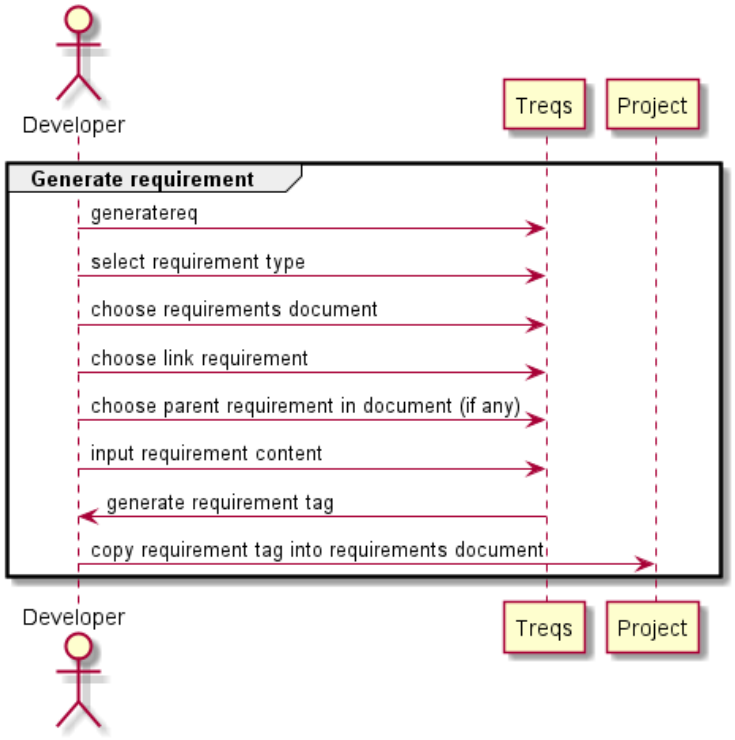


Fig. 7. Generate requirement workflow (an example of system requirement)

for each requirement. T-reqs creates a new file and writes these requirements with customIDs into the new file. The workflow is as shown in Figure 8.

3) Check consistency: Since each requirement should be traceable and unique, developers should make sure the consistency before uploading the changes of requirements documents into a version control platform. When developers execute the function Check consistency, T-reqs goes through all requirements documents that are under a given directory and tracing all consistency of requirements. There are three consistency rules of T-reqs: empty values tracing, duplicated values tracing, and links tracing. Figure 9 is the workflow of Check consistency: when the developer executes a command *treqs*, T-reqs requires the developer to specify which requirement types that the developer wants to check, and T-reqs requires developers to input the directory of requirements documents. Every time when T-reqs finishes a consistency tracing and finds tracing missing, T-reqs writes logs into a file as a summary of tracing check. If developers choose to check test case, in addition to these three consistency tracings, T-reqs also traces requirement tags that are in the common coding files.

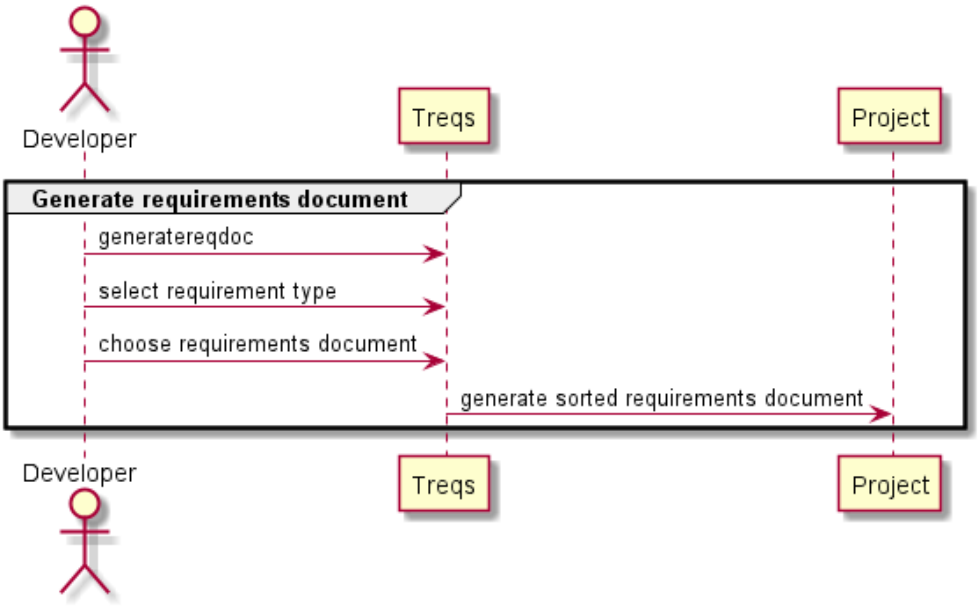


Fig. 8. Generate requirements document workflow

B. Requirements elements

Each requirement in a requirements documents is stored as a HTML tag format, which is presented as <element></element>. The basic fields of a requirement element tag are 'id', 'type', 'link_element', 'email', and 'date'. Within these fields, 'id' is a unique id of a requirement; 'type' is the requirement type; 'link_element' is the parent id that a requirement links to. If the value of 'link_element' is 'root', the requirement is at the first level in the requirements document (which customID starts from 1, 2, and so on). Since each system requirement should be traced to a user story, each system requirement element tag has an extra field 'link_us' that links to a value of 'id' field of user stories. On the other hand, each quality requirement should be traced to a system requirement, a quality requirement element tag has an extra field 'link_sr', which links to a value of 'id' field of system requirements. This is the same as test cases, which also has an extra field 'link_sr'. The following is an example of system requirement element tag:

```
<element id="437096961311370" type="SR" link_us="614396961311370"
link_element="root" email="person1@email.com" date="20200101 00:00:00"
>SR content</element>
```

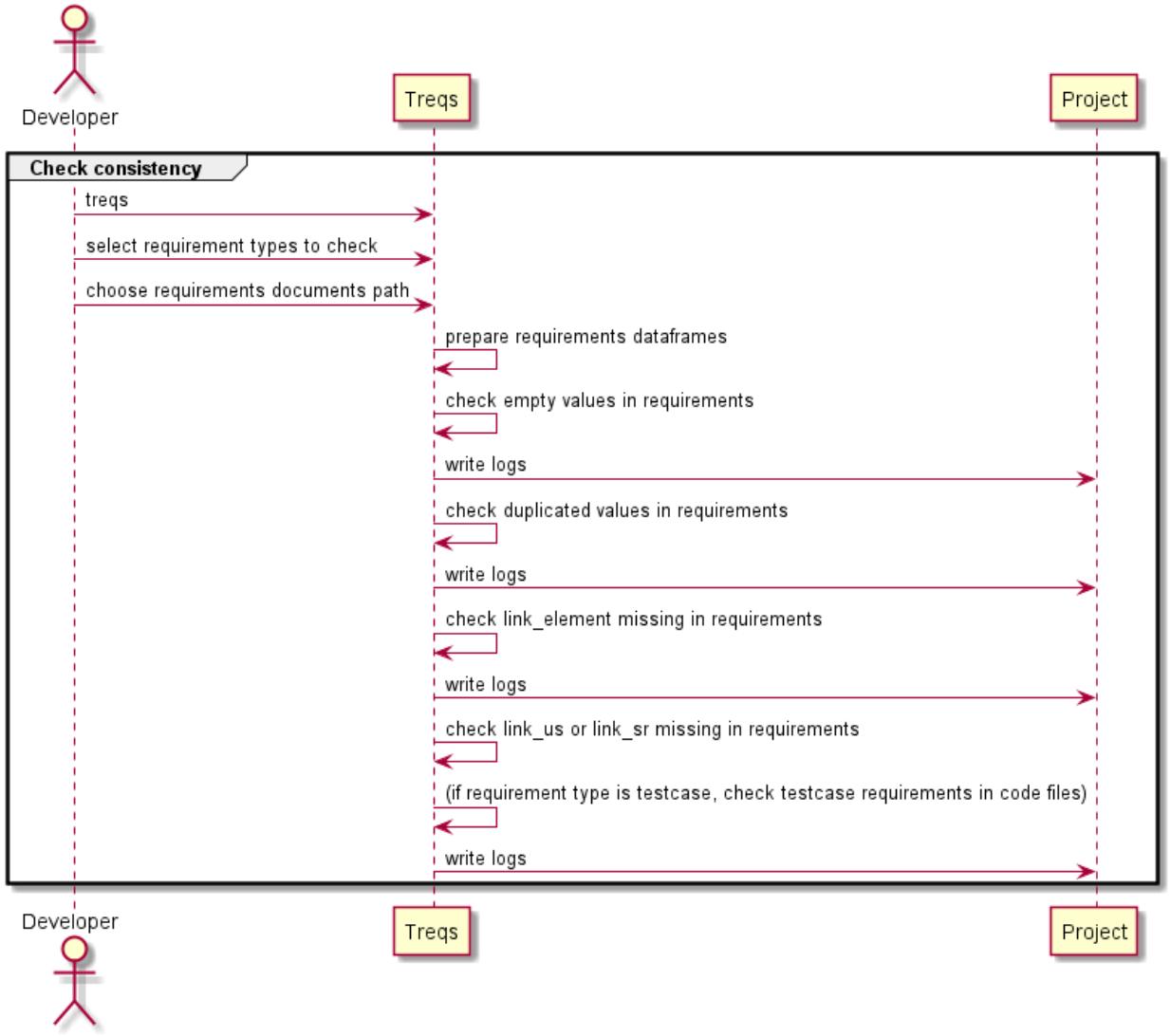


Fig. 9. Check consistency workflow

C. *Project classes*

The class Requirement is to store all commonly shared attributes of different requirement types. USRequirement, SRRequirement, QRRequirement, and TCRequirement are children classes, which inherit methods and properties from the parent class Requirement. The class UseRequirement is to manage all requirements objects. The scripts checkConsistency, generateReq, and generateReqDoc define T-reqs's functions Check consistency, Generate requirement, and Generate requirements document separately. The file parameters and commonFuncs define the parameters and common functions that are used in the whole

project. Figure 10 shows the relationship of each class.

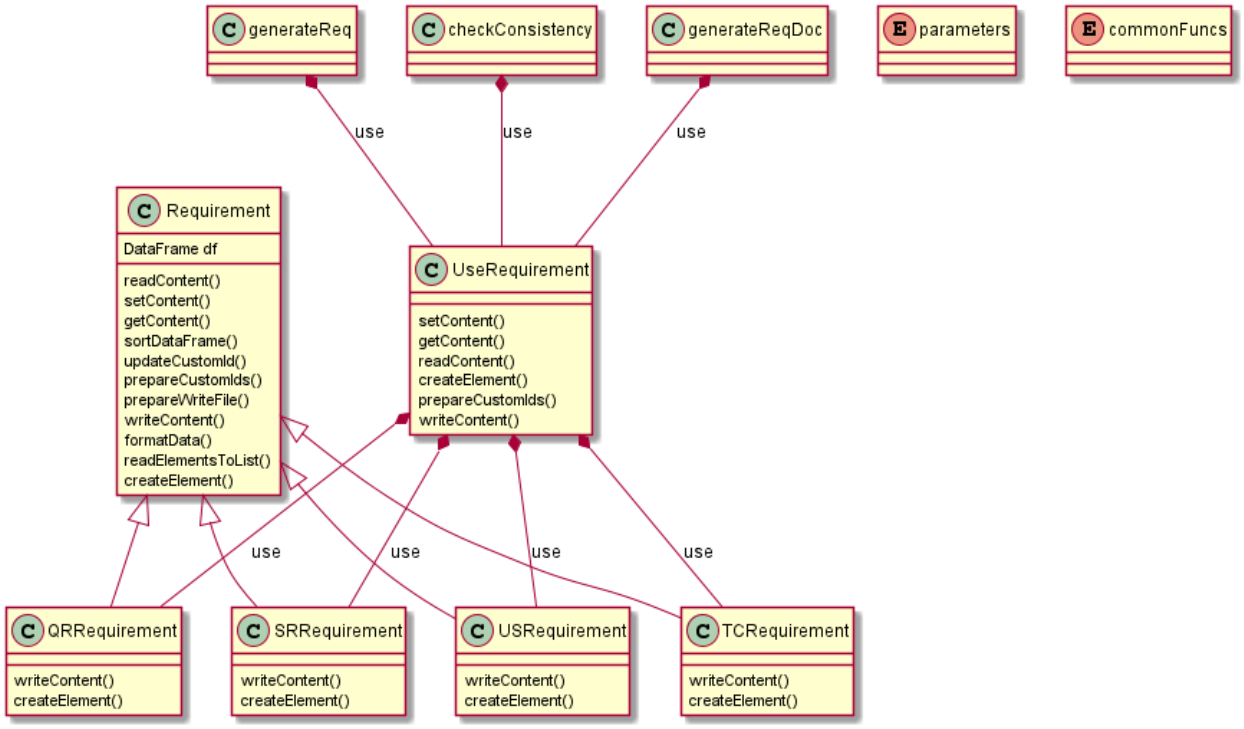


Fig. 10. Project classes

VI. Discussion

In this section, the main three features of previous T-reqs and new T-reqs are discussed.

A. Requirements IDs Management

The Amazon RDS solution ensures all generated requirements can have a new numerical ID based on the latest ID number. In other words, every time developers generate requirements, these newly generated requirements are stored into databases. However, once newly generated requirements are not accepted, the IDs of these requirements become incorrect and affect the value of ID when the next time developers generate new requirements. Besides, since a team has maintain the Amazon RDS to make sure these databases work correctly when using T-reqs, members of a team may spend extra effort and time on these Amazon accounts. Furthermore, this issue increases the complexity of using T-reqs.

Another solution is to use a unique ID and generate requirements with customIDs. In this solution, ids of requirements are stored and maintained in requirements documents,

which means there is no need of having remote databases to manage IDs. Hence, to compare with Amazon RDS solution, it is easier for developers and teams to manage requirements IDs. However, because numerical IDs are stored with an order in requirements documents, a developer has to update all related requirements in a requirements document when inserting a new requirement. As a consequence, due to the need of updating other relevant requirements, developers are not able to take their own responsibilities on editing requirements documents.

The solution which is adopting unique id and generating requirements without customIDs is proposed to solve the issues mentioned above. Since all requirements in requirements documents do not have number IDs, when the developer inserts new requirements into the requirements document, he or she does not need to update other requirements. The numerical IDs, which are customIDs are generated when developers execute the function of T-reqs. This solution allows members of a development team to take their own responsibilities on maintaining their own changes in requirements documents.

B. Functions and workflows

Generate requirement: when developers generate new requirements, the previous T-reqs provide the latest available IDs for developers. Since requirements should follow the same format in order to be managed, the previous T-reqs provides template requirements documents with requirements examples. Developers follow the same format when manually insert requirements into requirements documents. Different from the previous T-reqs, the new T-reqs creates the entire requirement element based on information fields provided by developers to T-reqs. Therefore, the new T-reqs reduces a step of generating requirements template for developers. All the newly created requirements follow the same format, which makes it easier to be maintained and managed for a team.

Generate requirements document: because in the previous T-reqs, all requirements include IDs in requirements documents, there is no extra work for creating a new requirements document for managing IDs of requirements. the function Generate requirements document that provides by the new T-reqs works as a preview function of requirements tool, which is used for creating requirements documents with sorted IDs for different stakeholders (such as developers and product owners). Generate requirements document reduces the steps

of managing IDs during the requirements generating phase for developers by sorting and adding all corresponding requirements IDs simultaneously.

Check consistency: the main difference of consistency checking of the new T-reqs and the previous T-reqs is that the new T-reqs traces all requirements in requirements documents with unique IDs rather than numerical IDs. Because numerical IDs change when requirements in requirements document are updated, when a numerical ID is incorrect in a requirements document, all the rest of the requirements in the same document loss traceability due to incorrect IDs. The unique ID of each requirement ensures all requirements can be tracked no matter the numerical ID that the requirement has.

C. Configuration management

In the previous T-reqs, it requires developers to provide a lot of information, for example, a path of requirements documents and file patterns of requirements documents. This information is written into a configuration file, and it is loaded and used when developers execute functions of T-reqs. This reduces the steps of developer inputting data and save developers' time, however, a drawback is that it lacks the flexibility of managing update (such as changing path of requirements documents). On the other hand, the new T-reqs does not store any information at the initialization phase. When developers execute functions of T-reqs, T-reqs asks developers to provide necessary information at the runtime. The new T-reqs requires developers to spend extra time on input information every time when executing functions, which has the disadvantage of time-consuming but has the advantage of handling any changes of configurations.

VII. Conclusion

The main contribution of the new T-reqs is that it simplifies requirements IDs management and provides a robust way of requirements taceability. Besides, the new T-reqs enables developers to create structured requirements through standardized HTML requirement element tags. However, due to the limited developing time, two functions that the previous T-reqs has are not included in the new T-reqs. These two functions are preview plantUML models and support multiple repositories, which should be included into the next version of T-reqs.

How to manage configurations of the next version of T-reqs should also be considered. Developers should only need to input runtime information for T-reqs when executing func-

tions, information that has less chance to be updated should be stored and reused. Thus, it is needed to define what kind of information that should be stored into a configuration file, and what kind of information is required whenever developers executing functions. Besides, sufficient instruction and interface should be provided for developers in order for them to update configurations in an efficient manner.

Since the T-reqs relies on using a well-built environment, this means that developers have to handle all the project settings to provide a viable environment for using T-reqs. Developers must install the Python environment manually before installing T-reqs. In addition, multiple libraries are required to support the project functionality, when a developer installs T-reqs, all libraries are installed on the developer's device. These project settings increase the complexity of using T-reqs, and they provide potential error creations, thereby preventing users from using T-reqs. Therefore, the next version of T-reqs should simplify the environment setting to make T-reqs easy to use.

REFERENCES

[1] Y. Alsahhar and G.-M. Bulai. Using t-reqs as teaching solution for requirements engineering in second cycle university courses. 2019.

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

[3] M. G. Gebremichael. Requirements engineering for large-scale agile system development: A tooling perspective. 2019.

[4] R. Kasauli, G. Liebel, E. Knauss, S. Gopakumar, and B. Kanagwa. Requirements engineering challenges in large-scale agile system development. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 352–361. IEEE, 2017.

[5] E. Knauss. Tool support for managing requirements in large-scale agile system development. *GitHub repository*, 2018. https://github.com/regot-chalmers/treqs.

[6] E. Knauss, G. Liebel, J. Horkoff, R. Wohlrab, R. Kasauli, F. Lange, and P. Gildert. T-reqs: Tool support for managing requirements in large-scale agile system development. In *2018 IEEE 26th International Requirements Engineering Conference (RE)*, pages 502–503. IEEE, 2018.