Non-Functional Requirement Traceability Process Model for Agile Software Development

Adila Firdaus Arbain[1], Dayang Norhayati Abang Jawawi[1], Imran Ghani[2] and Wan M. N. Wan Kadir[1]
[1]*Faculty of Computing, Universiti Teknologi Malaysia, 81310 Johor Bahru, Johor, Malaysia.*
[2]*School of Information Technology, Monash University Malaysia.*
adilafirdaus@gmail.com

***Abstract*—Agile methodologies have been appreciated for the fast delivery of software. They are criticized for poor handling of Non-Functional Requirements (NFRs) such as security and performance and difficulty in tracing the changes caused by updates in NFR that are also associated with Functional Requirements (FRs).This paper presents a novel approach named Traceability process model of Agile Software Development for Tracing NFR change impact (TANC). In order to validate TANC's compatibility with most of Agile process models, we present a logical model that synchronizes TANC with the two of enhanced models: secure feature-driven development (SFDD) and secured scrum (SScrum).Then, we conducted a case study on TANC using a tool support called Sagile. In terms of adaptability with agile process model, the logical model could be depicted in SFDD and the case study proved that TANC is carried out successfully in SFDD.**

***Index Terms*—Agile Methodologies; Feature Driven Development; Non-Functional Requirement; Scrum.**

I. INTRODUCTION

Most of the software teams deal with non-functional requirements (NFR) in an ad-hoc fashion. [1]-[3]. There is also a few discussions about implementing security [1] in Agile software development models such as Scrum [4],[5]. Some of the teams claim that they only trace the NFR if the software is a safety or security-based system like an e-banking system [5],[6]. The rest of the NFRs are just formality checks usually performed at the end of the development process [7],[8]. Tracing NFR in agile approaches becomes worse because the clients or users often ignore safety and performance but expect the system to be developed fast. In this hassle, the mishandling of NFRs brings fatal consequences to the software. Then, agile software development is seldom equated with the NFR measurement such as a secure development, due to lack of formal processes, understanding and emphasis on security instant issues [3]. Therefore, it is difficult to apply traditional security controls such as risk analysis, formal validation of internal and external security reviews while practicing Agile software development process. All these issues occur due to a number of reasons. For example, traceability principles [9], are more clearly defined in comprehensive time and heavyweight processes [10] compared to agile principles that are more flexible, easy and loose couple [5],[8],[11]. Both are two different principles. For an example, one of the NFR (security) [12],[13] verification and traceability is too redundant and documented wise that go against with Agile method disciplines and does not properly show the relation directly with the structure of the system. Furthermore, Bartsch (2011) states that neglecting communication and interaction patterns in agile development such as tracing security will lead to a loss of detection on security measures (authentication and operational security). However, if traceability could be well defined in terms of the procedures, and can be simplified, flexible, agile and manageable [13],[14],[16], then it can produce satisfactory results. By looking at each one of these attributes we can develop quality software by solving and tackling each of these problems with traceability.

In conjunction with the issues discussed above, the rest of the paper is organized as follows: Section II presents the Traceability process model of Agile Software Development for Tracing NFR change impact (TANC) and its mechanism, the phases inside the TANC process model and the integrated methods. Section II demonstrates the SAgile tool support and Section IV presents the evaluation process by using logical model and a case study that deployed TANC process model Lastly, Section V presents the conclusion of this study.

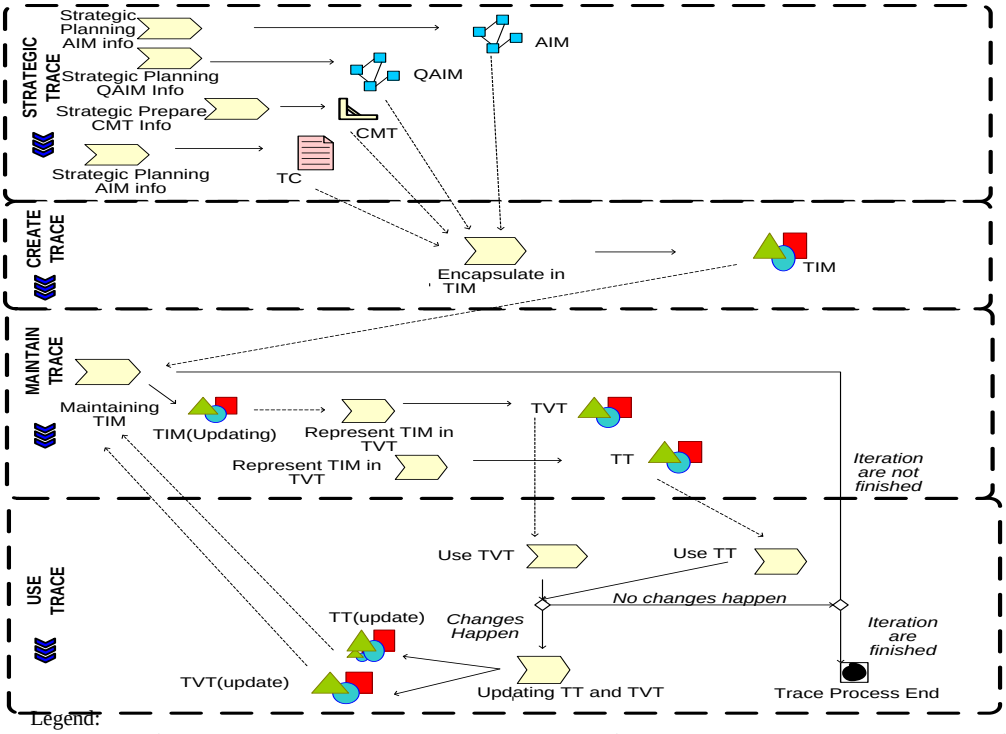
II. TANC PROCESS MODEL

Basically, the process model traceability has four main phases and each phase has its activity flow and techniques. Figure 1 depicts the decision on how to use the traceability in order to help trace the NFR change impact in agile software development. It starts with the strategic trace phase that does the planning of creating trace artefacts. In this phase, agile information management (AIM), quality agile information management (QAIM), Change Management Table (CMT) and test case (TC) are collected during the requirement elicitation process. AIM contains all the information such as user stories, backlog, iteration feature, timestamp and link information while QAIM holds the data on the NFR, NFR timestamps and the link information of the NFR. CMT is explained in Subsection D.

Lastly, TC is prepared after the requirements elicitation process and the NFR are well defined in the early stage of development. These TCs will be used as a trace indication to show any change impact that happens to the NFR if some FR are changed during the iterations. Thus, it is important to create both FR and NFR test cases. Then, during the create trace phase, all the trace artefacts and trace links are defined and stored in the traceability information management (TIM). After the create trace phase, the traceability is used again during iterations in the test phase to update the NFR based on the test cases. If changes happen and the NFR also need to be re-evaluated, then the next phase is use trace phase. In this phase, the traceability information storage is

represented in the form of TVT and TT. TVT and TT provide clear trace vision in order to see which user story and NFR are affected during the requirement changes. After some modification of the system during the next iteration, the maintain trace phase will update the TIM.

The next sections present each phase in details. For example, in the use trace phase, the information model uses TT to show the current evolution of the user story development. The strategic trace phase applies the quality attribute workshop (QAW) technique in planning the hierarchy and decomposition of NFR elements, as well as visualization of backlog, user story and all types of changes.



Legend:
AIM: Agile Information Management, QAIM: Quality Attributes Information Management, TIM: Traceability Information Management, TT: Trace TimeLine, TVT: Trace Version Table, CMT: Change Management Type, TC: Test Case

Figure 1: TANC process model

A. *Strategic Trace Phase*

The strategic trace phase is important for strategically planning and structuring the maintenance process, changing the impact information, changing the propagation and evolution of trace artefacts. One of the trace artefacts that are used in this phase is QAIM. Basically QAIM is derived from QAW (Quality Attributes Workshop) attributes. From the derivation of QAW attributes, this stage plans the links of related NFR into coarse-grained requirements.

The strategic trace phase presents how the trace links/relationships are drawn across the user stories to the NFR. This phase is highly important as it will determine the update of change impact during the development and testing phase. Figure 2 shows the activity diagram of strategic trace phase.

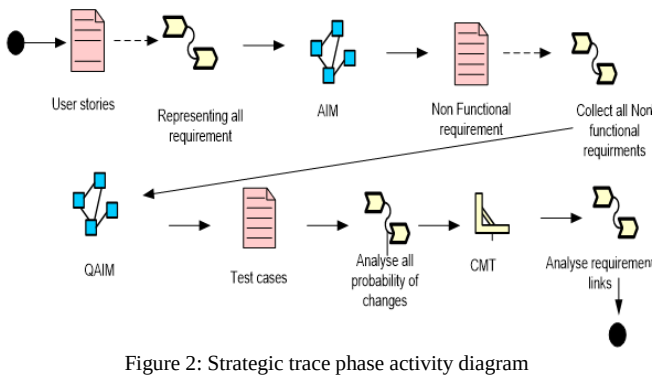


Figure 2: Strategic trace phase activity diagram

B. *Create Trace Phase*

The create trace phase comes after the strategic trace phase. In this phase, the development team or the modeling team will create the trace artifact that will be used to trace in the software development. Therefore, this phase must be done before the iteration starts in order to determine the set of requirements (set of sprints in scrum and set of features in FDD). Figure 3 shows the process flow in the create trace phase which creates the trace links based on the four components of trace artefacts that have been initialized and analysed during strategic trace phase, AIM, QAIM, CMT and TC. All these trace artefacts are the information form in TIM.

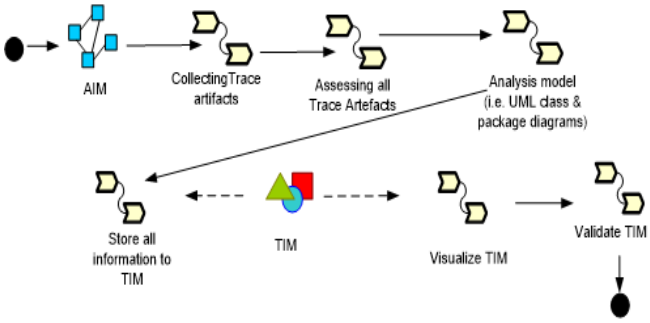


Figure 3: Create trace phase activity diagram

C. *Maintain Trace Phase*

The maintain trace phase helps to solve the propagation

issue in updating changes in requirements. This phase is crucial for preparing the trace artefacts that will be used during or after requirement changes phases. This phase accuracy should be determined by the representation trace in the create trace phase, but the method in this phase determines the consistency of the whole traceability. Therefore, this phase is quite important. This phase is divided into two process flows, which are maintaining trace n (normal requirement and NFR trace) and maintaining trace n.1 (updating the change of requirement). The maintaining trace phase is for normal updates of the test cases, NFR status and the user story. This trace has a tendency towards backward traceability techniques. In addition, maintaining trace for requirement change also has the tendency towards bi-directional traceability techniques, where the user story and affected user stories have to be changed first, followed by the NFR(if the change impact is reflected on the NFR) and lastly on the test cases. Next, the horizontal tracing technique applies tracing between related NFR. After updating each trace (propagate), the final result will be depicted in the traceability information storage. Figure 4 illustrates the steps in the maintain trace phase.

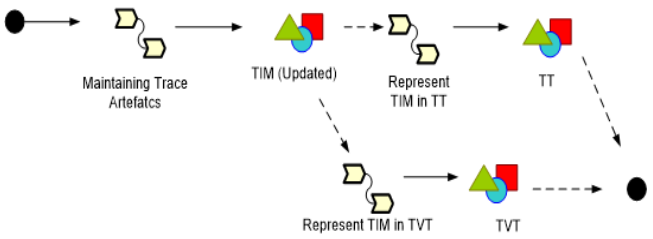


Figure 4: Maintain trace phase activity diagram

D. Use Trace Phase

The use trace phase is a conditional phase where this phase is only considered when there is a change in the requirements. The source of this phase is from the traceability information storage (TIM). The development team will recheck each relationship between user story and NFR, and NFR and NFR which being updated during the maintain trace phase. This phase will help the developer in making the decision to trace which user story and NFR if there are some changes made during the development process. It also helps the developer to check the progress of the development. The calculation of development progress will be done in future work. Lastly, it will help the developer to decide whether or not the iteration has been completed. This phase is recommended to use automated system that could generate the traceability graphs, matrixes and timelines. Figure 5 illustrates the activity diagram of use trace phase.

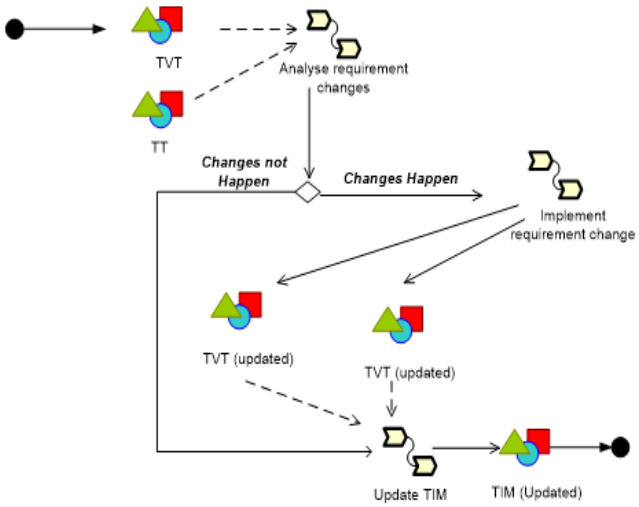


Figure 5: Use Trace Phase Activity Diagram

In TANC, we introduce a new representation form that is TVT to show the link between the FR and NFR for the many-to-many relationship (Table 1). They are represented by the versioning number in TVT (Table 1). After that, the trace artefacts in the traceability information storage is visualised using TVT. The orange highlighted box in Table 1 shows that the user stories that belong to backlog 1 (US 1.1) are linked to the access control (US 1.1.S.1) that belongs to security NFR. The versioning numbers show the link or relation between each component of the requirement and also show the layer of the requirement level (backlog, user story, iteration). The versioning label also shows the relationship among the NFR, as shown in the green highlighted box. (S.3.P.3) shows the relationship between security and performance.

Table 1
Trace versioning table

Iteration	Backlog Panel	User Stories Panel	FR→NFR (Test Cases)	QAW		
1	BG1	US1.1	US1.1.S.1 US1.1.S.2 US1.1.P.2	access control (S.1) Encryption (S.2)	Resistance	Security
2		US1.2	US1.2.S.1 US1.2.P.1 US1.2.P.2	Offline (P.1) Online (P.2)	Scheduling	Performance
3	BG2	US2.1	…	S.3.P.3		Security + Performance

Some researchers [16] identified types of changes of FR and how to deal with this issue by introducing event-based traceability and using the technique of subscriber and publisher. However, she overlooked the change impact of NFR for each change that has been applied on FR, thus she created goal-centric traceability (GCT). Nevertheless, this technique has its own weaknesses. It is unable to solve the scalability issue and cross-cutting issue that cause traceability redundancy. This technique also cannot be applied to the agile process because this technique is architecture-centric. Table 2 presents the types of changes that could impact NFR.

The symbols show the types of trace impact relationship in the traceability timeline. It can help the developer to determine which other potential NFR may change. It can also help in resolving the issue of redundancy of tracing

change impact on NFR. For example, the orange highlighted box in the table shows the possible changes if a user story has been modified. One of the impacts is on the security related to that modified user story, represented by the symbol "≥". There are also impacts on other NFR that affect the same or different NFR. For example, security to security uses the ↔ symbol and security to performance uses the S↔P symbol. These symbols are also used in the traceability NFR timeline.

Table 2
Change management table

Type of changes in FR (JaneClehuang, 2002)	Change impact to NFR (performance & security)	Symbol
Create New	Addition of new NFRs	→
Delete (--)	Deletion of NFRs	
Modified (-)	→Security	≥
	→Performance	
	Security↔Security	↔
	Performance↔Performance	
	Security↔Performance	S↔P
	Performance↔ Security	P↔S
Merge (++)	Security + Security=newSecurity	++
	Performance + Performance=newPerformance	
	Performance1= Performance2	
	Security1=Security2	
	Addition of New NFR	
	Deletion of NFR	
Decompose(+-)	Addition of New NFR	
	Performance1= Performance2	
	Security1=Security2	
	Deletion of NFR	
	Security=newSecurity + newSecurity	+-
	Performance= newPerformance+ newperformance	

The most suitable type of traceability representation form for evolving tracing is timeline format. The timeline is the best and simplified version of how to show the evolution of changing requirements. Therefore this study decide to use this representation form as one of trace artefacts representation, which called as traceability NFR timeline (TT). It could shows the update of changing requirements and the results of tracing the relationship of the user story to the NFR and the NFR to NFR. The timeline is depicted in Figure 6.

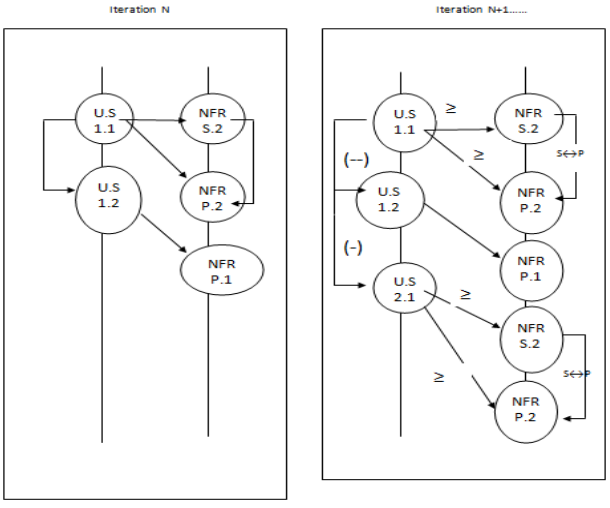


Figure 6: Traceability NFR timeline

The timeline shows the changes of each user story and NFR that already existed in each iteration. The timeline also shows the relationship between the user stories with other user stories and user stories with NFR. This relationship will be updated during the maintain trace phase while finishing each iteration. This relationship is very important to help the developer checking if any requirement changes. The function of each symbol is presented in Table 2. If changes happens, this relationship will help developers to know which user stories and NFR will be affected.

III. SAGILE TOOL SUPPORT

One of the objectives of this research is to develop a tool that can support the process of improved SFDD process. Figure 7 shows the main page of the SAgile tool.

Figure 7: Login Page

This tool has four main types of users based on the roles listed in FDD, namely project manager, chief manager, tester, and lastly the new role, master security. When each of these users logs in, they will see the project list. The project list records the systems that they plan to develop. Figure 8 portrays the features list in a project that has been added by the project manager or feature team based on client requirements. If the user clicks on a feature, the system will provide the details of the feature as depicted in Figure 9.

Figure 9 shows the details of 'check out' feature such as the estimated date, start date, and finish date. Each features is assigned to certain chief programmer and tester by the project manager. One of the speciality of using Sagile tools is it can assign specific security feature to the functional feature by the security master role. Based on Figure 10, 'Make Booking' feature is highlighted in red.

This shows that this feature has embedded security feature as depicted in Figure 10. Based on this figure, 'Make Booking' feature is linked with SQL injection and XSS features as those security features' checkboxes are checked. After this, a statistical analysis is conducted on the logical model for both SFDD and SScrum models. This analysis is conducted to evaluate the relevance of these logical models towards this research.

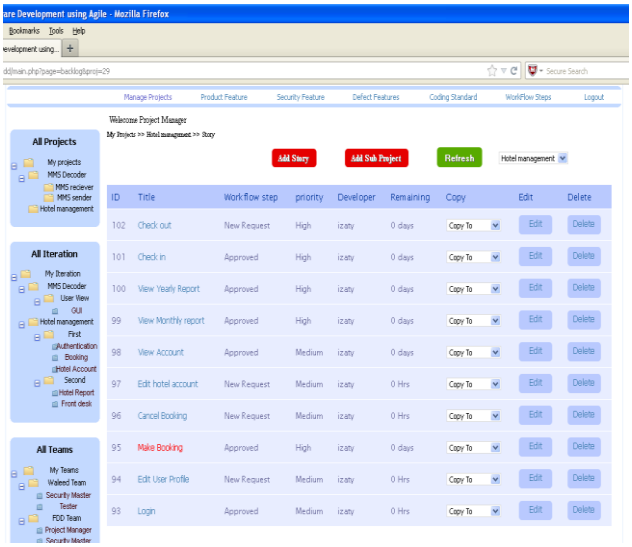


Figure 8: SAgile features list

Figure 9: Example of a feature's details

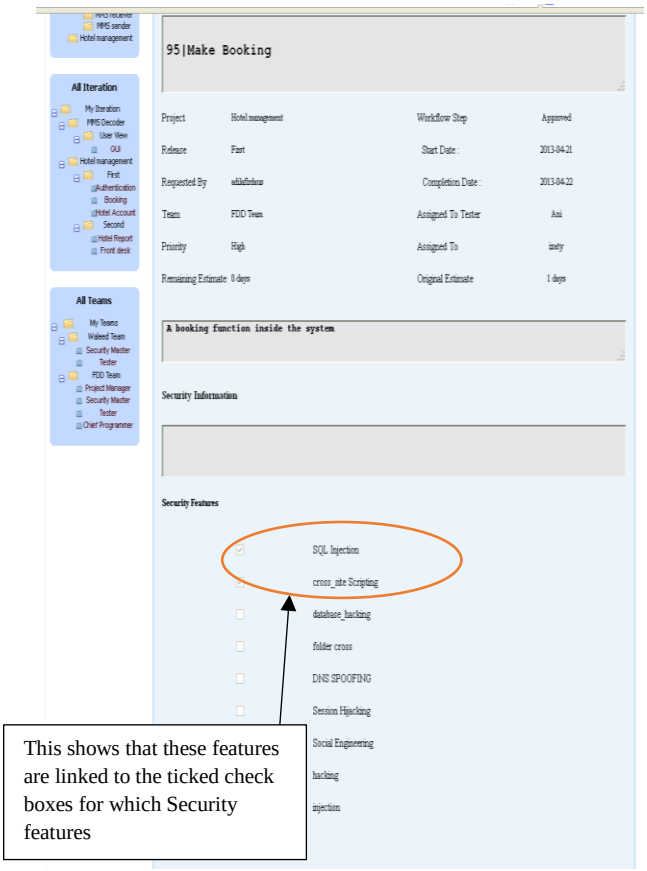


Figure 10: Interface of Make Booking feature

IV. CASE STUDY

This section presents the preliminary evaluation of the TANC process model in the case study. The discussion includes the design of a logical model in synchronizing TANC with security improved Scrum(S-Scrum) and FDD (SFDD). This logical model is the instruction of using TANC in both process model of SFDD and S-Scrum. Then, a case study applying the logical model have been conducted. For the case study, we use the SFDD logical model in order to show how the TANC process model will be used in the actual enhanced FDD process (SFDD).

A. Agile Trace Logical Model

This section presents a description of each trace phase that can be synchronized with the agile process models in the logical model manner. However, in this logical model, we do not try to synchronize with a normal agile process model but with an enhanced security agile process model, namely, the SScrum [17] on Figure 11 and SFDD [18] on Figure 12.

The strategic trace phase will be done during the product backlog collection phase, and the create trace phase will be done in the sprint backlog planning meeting phase. As the sprint iterates, the maintain trace phase will iterate as well and the use trace phase is used during the daily meeting process. Figure 12 shows the synchronization of the traceability phase with SFDD.

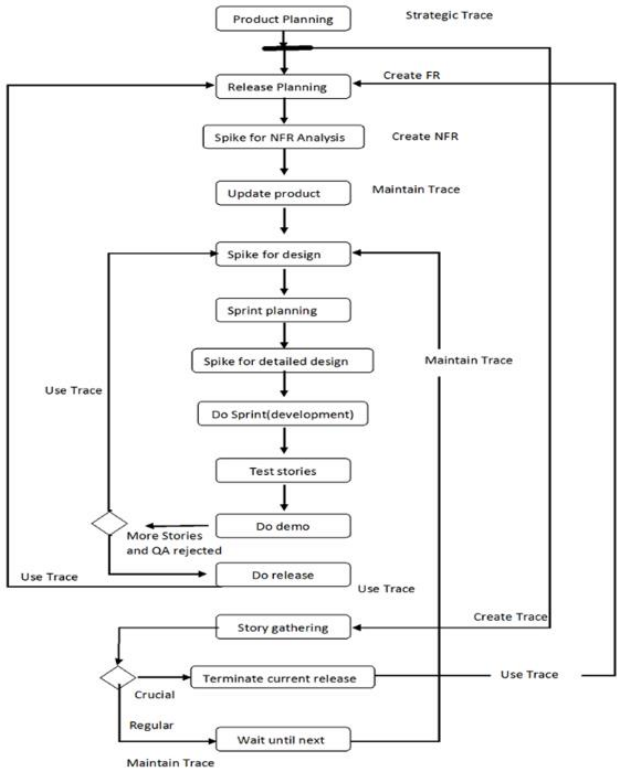


Figure 11: SScrum logical model

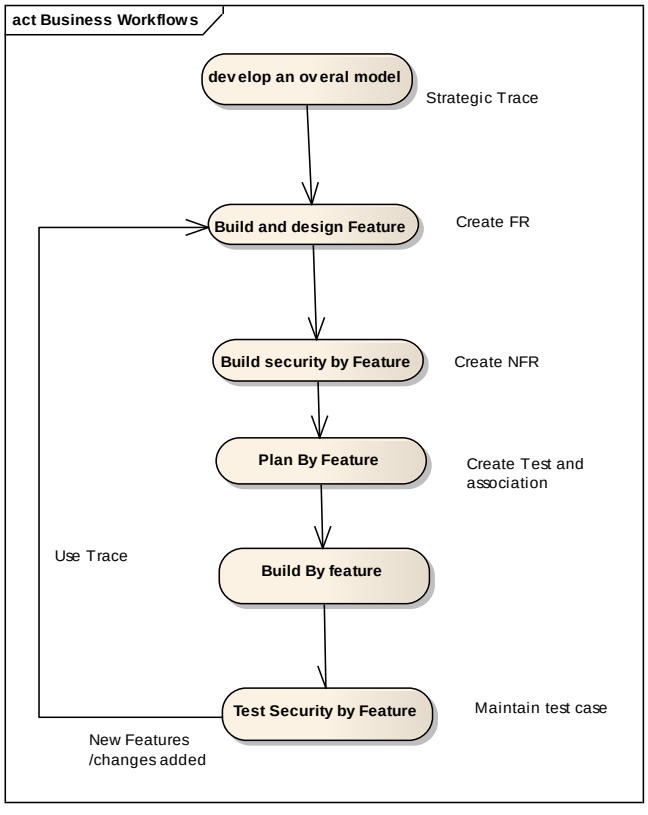


Figure 12: SFDD logical model

Based on Figure 12, it shows that the strategic trace phase will be done in the development of an overall model phase and the create trace phase will be done during the build a feature list phase and the plan by feature phase. As it iterates between the design by feature and the build by feature, the maintain trace and the use trace phases will iterate as well.

B. Hotel management system case study experiments using TANC

The hotel management system has 26 features and some of the features were filled in with the SAgile tool as in Figure 13. These features were filled in, in the form of user stories. Therefore, all the information about each feature management linked together in a page. This case study is explained based on the order of SFDD and TANC phases.

C. Strategic trace and create trace phases in developing an overall model

The starting phase in SFDD is called as the Develop an Overall Model phase. During this phase, the TANC strategic trace phase starts concurrently. This phase will start collecting all the related system features such as AIM, QAIM CMT and TC. All these feature trace links and connection will be planned out during this phase.

Figure 13: Features listing in Hotel Management System Project

Based on Figure 13, it shows the feature listing of AIM and QAIM. The red fonts features are the indication of links between the AIM and QAIM and the blue fonts features are not linked to any QAIM features. This action are done during this two phases.

D. Build and design feature phase

Then we move to the next phase in FDD that is the build and design feature phase. In this phase, each feature is filled in with more detailed information including the duration, the team member and the tester.

E. Create trace phases in the build security by feature phase

As FR or AIM features are already considered, next we need to fill in the quality attributes information management artefact. As only security and performance are within the scope of this research, the QAIM section is shown in Figure 14 listing all the security features and in Figure 15 listing all the performance features. Based on the new enhancement of the FDD model, this phase is specifically handled by the

Security Master that keeps track of the quality of the system especially the security features.

Figure 14: List of security features

After building the QAIM, the next step is to link each feature directly to each AIM feature as shown in Figure 16 using the SAgile tool. The red highlighted box shows that the security and performance details for the assigned feature while the two green highlighted boxes show security and performance features were chosen to link with the feature. The security and performance details are very important for the developers to code appropriately based on the requirements and the tester to test exactly based on the requirements. Consider it as extra notes for the developer and the testers. This tool's features are mapped from the techniques of TVT whereby it lists out the granularity from each AIM and QAIM and then map the links from the lowest level of granularity.

Figure 17 shows the view from the QAIM side. It shows which AIM features are attached for each quality feature and the status of the development of the parts of the system. This shows that the TANC approach applies bi-directional traceability techniques.

Figure 15: List of performance features

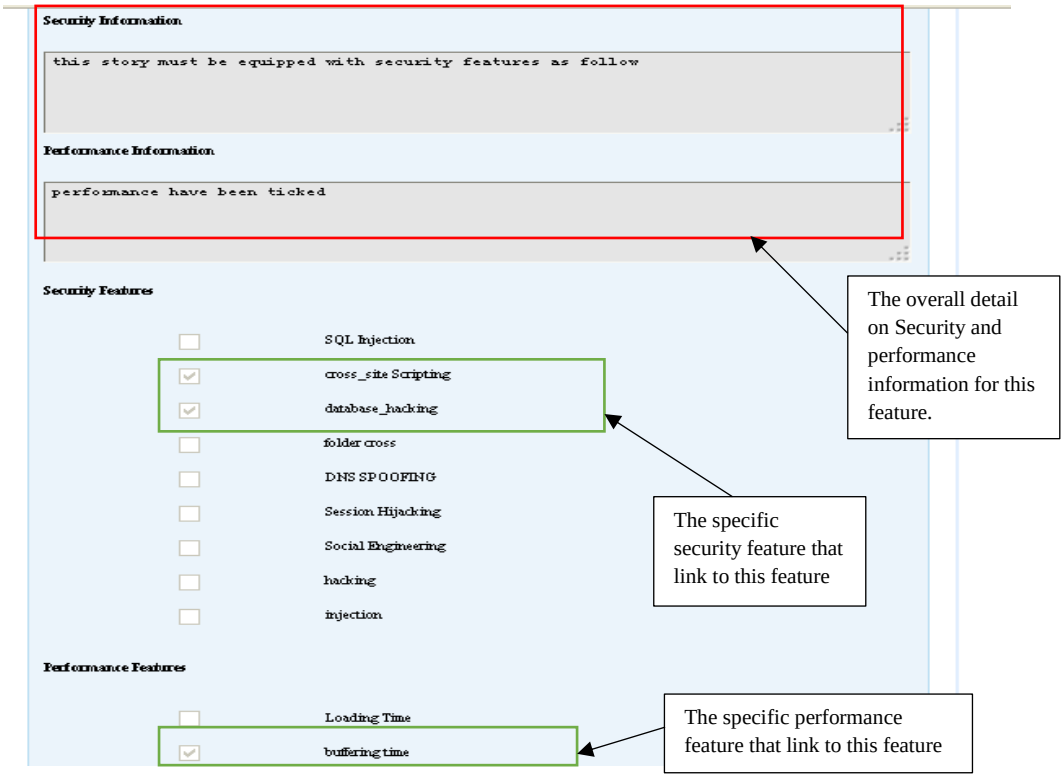


Figure 16: AIM and QAIM features links

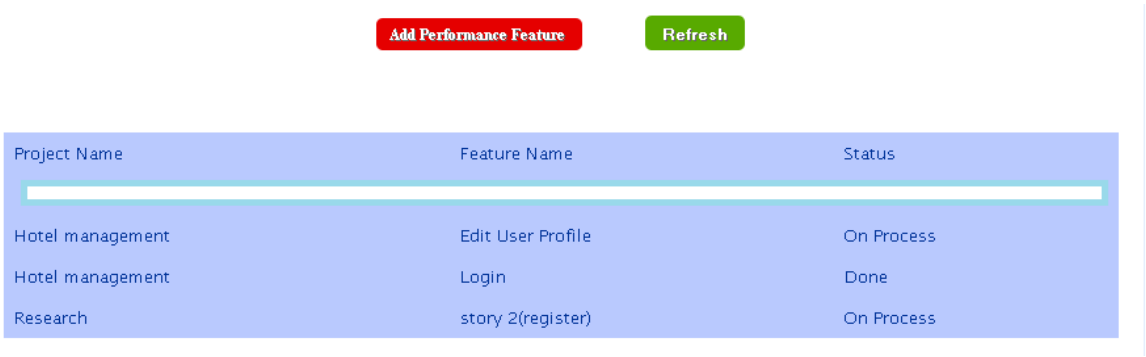


Figure 17: QAIM status based on project traceability table

F. Maintain trace and use trace phases in plan by feature phase

As shown in Figure 18 depicts the iterations listing that have been set during plan phase and Figure 19 shows the set of user stories that linked under one of the iterations. This is the phase where all the features are placed in each iterations. In Figure 19, the green highlighted box shows that the feature is in green font. This means that the feature has been assigned to both security and performance features. This phase is closely associated with the maintain trace and use trace phases. As the iteration starts to incrementally iterate, the maintain trace phase also runs simultaneously; however, in this case, the SAgile tools helps to automate the process of this trace phase.

Figure 18: Iteration feature listing

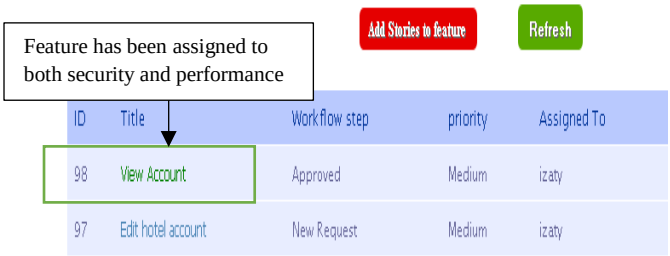


Figure 19: Features listing by iteration

Then, the use trace phase is used when the features need to be re-evaluated based on the changes needed and the changes have happened during the development. In the plan by feature phase, the use trace phase is used in order to plan the arrangement of each phase if needed. For example, a feature is done from the previous iteration but suddenly that feature needs to be improved. Therefore, it needs to be arranged in the next iteration. In order to track which features and iterations have been changed, the use trace phase is used. During this phase, the traceability information model that has been mapped in TVT and TT is formed, allowing us to track and document any change impact of the feature especially toward quality features.

G. Build by feature phase

In this phase, all the documentation is coded to build the features of the overall system. Even though this phase does not relate directly to any traceability phase, this phase is like a middle process in order to make the traceability as light as possible. This is due to the reduction of the trace artifacts.

H. Use trace phase in the test security by feature phase

During this phase, every feature that has been built will undergo testing. The current agile process usually only starts testing the AIM features and the QAIM features. These features are only tested when the whole system is completed. However, in the enhancement model, quality assessment testing must also considered especially in every iteration. The results from this testing are the input in the TIM as shown in Figure 20, during the use trace phase. In this way, the TANC approach can track the change impact toward any changes and the changes propagated from the features toward the quality features of the system.

When the development team has some changes or additions to the current system, they use the TVT or TT in order to check the status of the system. CMT helps to symbolize any type of changes that happen between the AIM features and QAIM features. This will help the development team to easily track what type of changes and which parts of the system are affected after applying some changes.

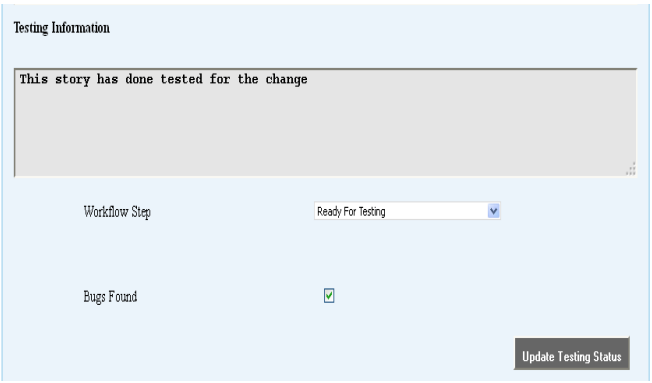


Figure 20: Feature's test result

Since agile practices involves less decoupling between features, it is easier to act on them as individual units and because of this if any changes happen, it most probably does not affect the other features. After a few iterations, if

changes happen the developer can spot which specific features are changed because by using TVT, it has decomposed NFR as individual units that are directly related one-on-one with FR plus with other related NFR. This reduces the effort involved in the trace because it will be easy to find which parts of the system are affected. Since the system is built increment iteratively order, when changes happen during the development it could easily modify the design of the system. However, a method of presentation that can easily show the evolution of the system development is needed. Therefore, in this study, we represent it in timeline format. The timeline presentation format is able to show the evolution of the development based on which iteration it is on, so that we can know which part needs to be traced.

V. CONCLUSION

This research paper has reviewed the current issues terms of NFR traceability and agile methodology and to solve the issues that have been highlighted. In conjunction with the issues discussed above, this paper presents a new traceability process model, TANC, that is consists of the traceability process and improved techniques from the matrix table approach that is widely used in the traceability process. In this paper, we present how this new traceability process could be adapted with one of most commonly used agile methodologies, FDD. It is worth mentioning that this model have been improved with NFR management, with a special emphasize on security. Based on the case study shown in Section IV, TANC process model is proved to assist FDD in tracking the change impact of NFR by using SAgile tool support. Therefore, it is called as SFDD process model.

ACKNOWLEDGMENT

We are thankful to Universiti Teknologi Malaysia (UTM) and Ministry of Science, Technology and Innovation (MOSTI), Malaysia for funding this project under Vot No: 4S113.

REFERENCES

[1] B. Ramesh, L. Cao, and R. Baskerville, "Agile requirements engineering practices and challenges: an empirical study," *Information Systems Journal*, vol. 20, no. 5, pp. 449-480, 2010.

[2] R. B. Svensson, M. Host, and B. Regnell, "Managing quality requirements: A systematic review," in *36th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)*, 2010, pp. 261-268.

[3] J. Walden, C.E. Frank, and R. Shumba, "Teaching software security with threat modeling," *Journal of Computing Sciences in Colleges*, vol. 22, no. 1, pp. 119-120, 2006.

[4] J. Walden, C. E. Frank, and R. Shumba, "Teaching software security with threat modeling," *Journal of Computing Sciences in Colleges*, vol. 22, no. 1, pp. 119-120, 2006.

[5] M. Serrano, and J.C.S. do Prado Leite, "Capturing transparency-related requirements patterns through argumentation," in *Requirements Patterns (RePa), First International Workshop*, 2011, pp. 32-41.

[6] D. Gregorio, "How the Business Analyst supports and encourages collaboration on agile projects", in *2012 IEEE International Systems Conference SysCon 2012*, 2012, pp. 1-4.

[7] E. Hadar, and G. M. Silberman, "Agile architecture methodology : long term strategy interleaved with short term tactics," in *OOPSLA Companion '08 Companion to the 23rd ACM SIGPLAN Conference on Object-oriented Programming Systems Languages and Applications*, 2008, pp. 641-651.

[8] K. Mohan, P. Xu, L. Cao, and B. Ramesh, "Improving change management in software development : Integrating traceability and software configuration management," *Decision Support Systems*, vol. 45, no. 4, pp. 922-936, 2008.

[9] J. Cleland-Huang, O. Gotel, and A. Zisman, *Software and Systems Traceability*. London: Springer, 2012.

[10] M. Mirakhorli, and J. Cleland-huang, "Tracing Non-Functional Requirements," in *Software and Systems Traceability*, J. Cleland-Huang, O. Gotel, and A. Zisman, Eds. Berlin, Heidelberg: Springer, 2012, pp. 299-320.

[11] J.S Persson, L. Mathiassen, and I. Aaen, "Agile distributed software development: enacting control through media and context," *Information Systems Journal*, vol. 22, no. 6, pp. 411-433, 2012.

[12] S. Bartsch, "Practitioners' perspectives on security in agile development," in 6th *International Conference on Availability, Reliability and Security (ARES)*, 2011, pp. 479-484.

[13] M. Cardinal, *Addressing Non-Functional Requirements with Agile Practices Who Am I ?*. Agile Specification, Addison-Wesley, Spring 2012.

[14] J. Cleland-Huang, M. Rahimi, and P. Mäder, "Achieving lightweight trustworthy traceability," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2014, pp. 849-852.

[15] M. Qasaimeh, and A. Abran, "An audit model for ISO 9001 traceability requirements in agile-XP environments," *Journal of Software*, vol. 8, no. 7, pp. 1556-1567, 2013.

[16] J. Cleland-Huang, and D. Schmelzer, "Dynamically tracing non-functional requirements through design pattern invariants," in *Workshop on Traceability in Emerging Forms of Software Engineering, in conjunction with IEEE International Conference on Automated Software Engineering*, vol. 10, 2003, pp. 1.

[17] Z. Azham, I. Ghani, and N. Ithnin, "Security backlog in Scrum security practices," in *5th Malaysian Conference* in SoftWare Engineering, 2011, pp. 414-417.

[18] A. Firdaus, I. Ghani, and S.R. Jeong, "Secure feature driven development (SFDD) model for secure software development," *Procedia-Social and Behavioral Sciences*, vol. 129, pp. 546-553, 2014.