Jurnal
Teknologi

Full Paper

NON FUNCTIONAL REQUIREMENTS (NFRS) TRACEABILITY METAMODEL FOR AGILE DEVELOPMENT

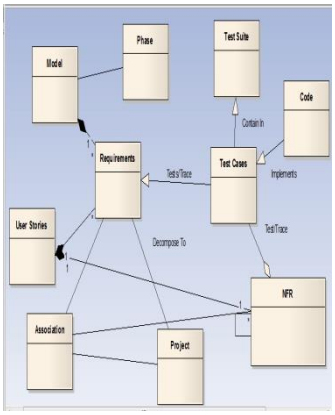
Adila Firdaus*, Imran Ghani, Dayang Norhayati Abg Jawawi, Wan Mohd Nasir Wan Kadir

Department of Software Engineering, Faculty Of Computing, Universiti Teknologi Malaysia, Malaysia

Article history
Received
2 February 2015
Received in revised form
8 October 2015
Accepted
12 October 2015

*Corresponding author
adilafirdaus@gmail.com

Graphical abstract



Abstract

Agile methodologies are well known for early and frequent releases. Besides, these methodologies also handle requirement changes well without causing delays. However, it has been noticed that the functional requirements changes can affect the non-functional requirements (NFRs) such as security and performance. It is also possible that the agile team is not even aware of these effects causing dysfunctional system. This issue could be addressed by offering traceability mechanism that helps to trace the effect of functional requirement changes on the non-functional requirements. Unfortunately, a few researchers have conducted studies regarding this issue. Thus, this study attempts to present a Traceability Process Model (TPM) to tackle the issue of tracing NFR especially security and performance. However, to materialize a full scale TPM, a metamodel is necessary. Therefore in this paper, we present a metamodel by integrating two existing metamodels. Then we validate the newly built metamodel with precision and recall methods. Lastly, we also develop a traceability tool that is based on the proposed metamodel.

Keywords: Agile methodologies, security, performance, traceability, meta model, propagation

Abstrak

Kaedah Agile terkenal dengan awal dan kerap pengeluaran. Selain itu, ia juga mengendalikan perubahan keperluan berfungsi dengan baik semasa pembangunan perisian tanpa menyebabkan kelewatan. Walau bagaimanapun, perubahan keperluan berfungsi boleh menjejaskan keperluan tidak berfungsi (NFRs). Hal ini mungkin berlaku kerana ahli kupulan pembinaan perisian tidak menyedari kesan-kesan ini menyebabkan sistem tidak berfungsi dari segi keselamatan dan prestasi. Isu ini boleh ditangani dengan menawarkan mekanisme pengesanan yang membantu untuk mengesan kesan perubahan keperluan berfungsi kepada keperluan yang tidak berfungsi. Malangnya, hanya terdapat beberapa orang penyelidik yang menyediakan kajian mengenai isu ini. Oleh itu, kajian ini bertujuan untuk membentangkan Model Proses Kebolehkesanan (TPM) untuk menangani isu mengesan NFR terutamanya dalam keselamatan dan prestasi. Walau bagaimanapun, untuk mewujudkan skala penuh TPM, metamodel TPM hendaklah dibuat terlebih dahulu. Oleh itu dalam kertas ini, kami akan membentangkan metamodel baru dyang berintegrasi yang dari dua metamodel sedia ada. Kemudian kami mengesahkan pembinaan metamodel baru dengan kaedah ketepatan / ingat. Akhir sekali, kami membangunkan alat dan dipetakan dari metamodel itu.

Kata kunci: Kaedah Agil, keselamatan, prestasi, pengesanan, meta model, perambatan

1.0 INTRODUCTION

Traceability has worked greatly in traditional software development process such as waterfall [1], model driven [1] and started to grow in Agile software development [2] too.

Traceability in software development process specifically on Non Functional Requirement (NFR) could be back tracked since the year of 1996 [3]. As far as NFR in agile development is concerned, recent work in 2012 [4], 2013 [5] and 2014 [6] have been done. The recent research has helped to solve a number of problems that always arise in software development process such as tracking the progress of the system development [7], and process improvement [8]. After doing literature review on the traceability and Agile projects, there are still a few main issues. One of the main problems in tracing NFR in Agile projects is that the change in functional requirements (FRs) impact on NFRs. However in this study, we only focus on two main sub issues which are propagation and inconsistency issues.

Propagation issue [9-12] is discussed in relation to model based or object-oriented [13] that leads to redundancy of traceability process in Agile environment. It means that the propagation techniques in most existing traceability include heavy weighted documents, time consuming and repeatable flows. Then the inconsistencies are important to make sure when changes happen. It could help the team to track back which requirements are affected. It also must prove how adaptive is the traceability technique when the new requirements are added to the existing system. As mentioned earlier, in this study we consider security and performance and their relation to propagation and inconsistency. Meanwhile, security and performance have different criteria and attributes inside the artefact to be traced. Due to that, propagation of change must be consistent and cover the whole different path. All these scenarios depict the research problems that need to be solved.

In order to address this issue, there are many traceability models [14], concepts [15] and mechanisms [16] that have been proposed in relation to NFR but none of them are compatible with Agile projects.

In order to develop a suitable approach, we develop a metamodel that supports the approach. This metamodel is explained in detail in this paper. In the next section we introduce a generic Agile traceability model (ATM) that becomes the base of traceability approach in any traceability model in Agile development process. In Section 3, we explain the NFR traceability metamodel (NFRTM). While Section 4 and 5 present the design of Agile NFR traceability metamodel as the result of two metamodels integration that have been explained in Section 2 and 3. The validation of the metamodel is also presented. Lastly, a tool was built in order to support this metamodel. This mapping of tool with the metamodel is explained in Section 6.

2.0 AGILE TRACEABILITY MODEL

The most common tracing scenario for an agile project is depicted in the ATM [17] shown in Figure 1. It is interesting to note that ATM has appeared as a result of the discussions with agile developers and therefore it reflects developers' perception of a common practice. It shows how traceability in ATM was first established between acceptance tests and user stories by inserting a reference to one or more users' history in each of the acceptance tests. This ATM is elaborated, and is supported by a number of efficient management tools such as Rally Software [17].

In ATM, testing is a way of tracing. During testing scenario, when the test cases are executed and passed, the developer confirm that the code implements the test [18], and therefore implicitly train the tracking "tools". This however means that the code is treated as a single artifact, without visibility associated with the test case class. This raises the question whether the level of granularity could be supported for the traceability purposes of impact analysis. In order to think about it, we need to know that how to determine which traceability links are used to identify potentially impacted sections of the code in order to plan a proposed change, manage risks, or estimate effort. However, the level of support required is not necessarily the most agile projects, especially if the project is small or moderate. This method is feasible unless the size of the project is too large, or the lifetime of the project and staff means that collective knowledge is sufficient for tracking.

In other context, one can show the suitability of the product because of the acceptance tests of the history of individual user. That individual users can identify whether they are satisfied with the code [19]. However, ATM cannot support the traditional impact analysis to identify areas of potential that lie within the code traceability links to plan a proposed change, risk management, and assessment efforts. One of ATM advantages is that it provides a very flexible mechanism anchored around traceability and user acceptance testing, and does not become brittle over time.

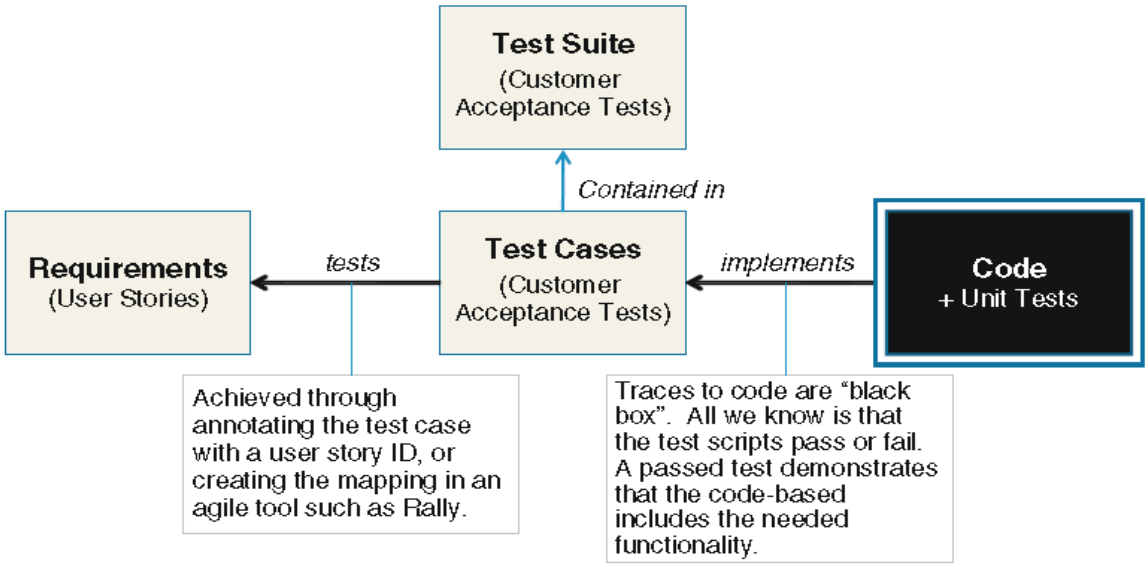


Figure 1 Example Agile Traceability Model [17]

3.0 NFR TRACEABILITY METAMODEL

NFRs can be traced at different stages in the project life cycle, and work within and across stages of the life cycle [20]. In order to be clear about the traceability of NFRs during the software development process it is necessary that the NFRs and their relations are explicitly modelled. A UML based NFR Traceability Metamodel (NFRTM) in Figure 2 shows the relations of requirements in scope [19]. In the model, NFRs are used as part of a group that is part of the model. The left side of the model, model application, shows the FR through the various stages of development. Each model is an aggregation of one or more artefacts for example use case diagram and a use case model, using a range model diagram and sequence diagram to model system analysis, communication diagram, class diagram and artifact such as a class, association, inheritance and class diagram. Artefacts and components of the model in this form give us the option of decoupling the work of tracing NFRs act or instance specific development. Right part of the NFRTM is the model that is used to model the hierarchy of NFRs and their relations. The decomposition of the NFRs is supported by non-functional models and can be achieved by using the goal-driven approach [21]. The items below show the explanation of each element in the metamodel.

- **Association**: an element that shows action or tasks of the association of **NFR**s with other elements such as FRs, Projects and Phase.
- **FR** (every element belonging to the **Requirement Group** modelling capabilities built in each elements of **Phase**): This refers to the practice field during requirement development.
- **Elements**: This refers to the foreign entity of NFRs. An example of such NFRs would be kept to 2 years as experienced **artifacts** in Oracle database software [19].
- **Project**: This refers to NFRs which provide a precise context to the project or development process. They are decomposition of NFRs, operationalization that refines the NFR into solutions in the system that will satisfice the NFR and Interactivity of NFRs.
- **Stakeholder**: element that guide the choice of NFRs associated with the FRs with which the parent NFR is associated. It also required to determine the existence of the relation between the **Requirement Model** elements.
- **Artifacts**: It gives the option of decoupling the task of tracing NFRs from a specific development practice or paradigm. For example, a use-case diagram for the use-case model, a domain model diagram and a system sequence diagram for the analysis model, a class diagram and a communication diagram for the design model.
- **Requirement model and Requirement Group**: FRs and NFRs are modeled as parts of a requirements group which is a part of a requirements model.

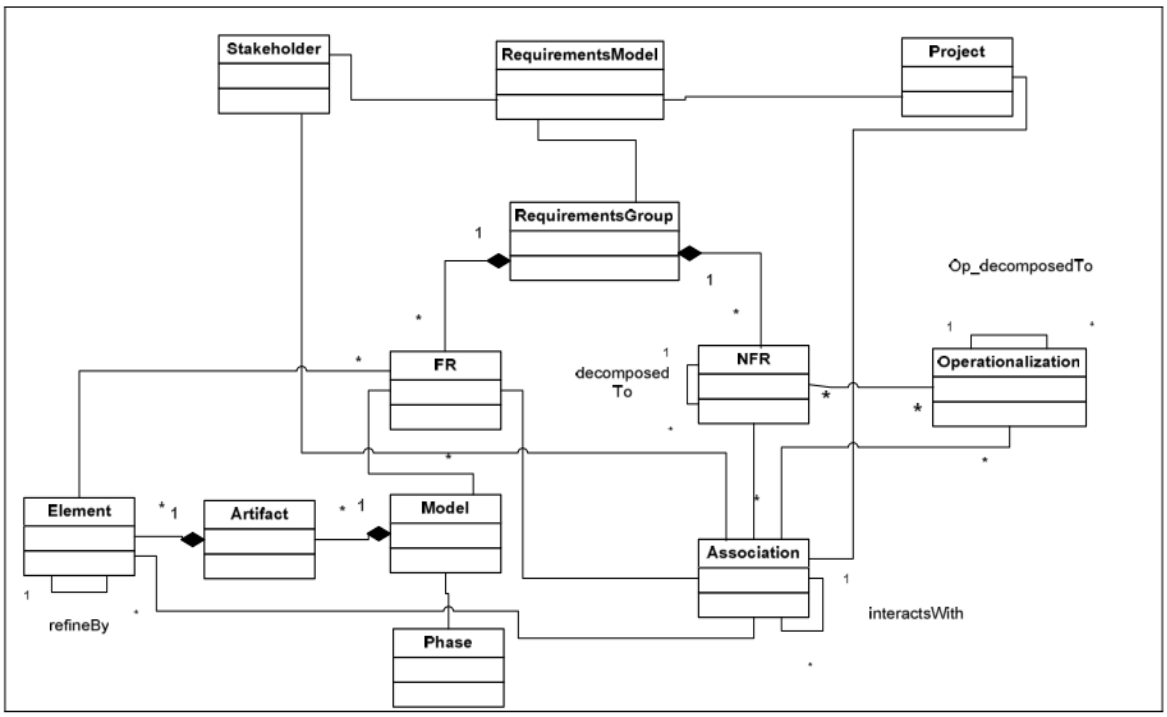


Figure 2 Example NFR Traceability Model [20]

4.0 AGILE NFR TRACEABILITY METAMODEL

In this section, an integration of agile and NFR traceability metamodel will be explained in detail. This integration formed a new metamodel that is Agile NFR Traceability Metamodel that has been specifically modified for resolving issues in this research area. As shown in Figure 3, the user stories, requirements (FR) and non-functional requirements (NFR) are decomposed from requirements elements. These three elements are linked during software development process. For example, a fast response time NFR associated with the order that the system should have the ability to accept orders during run time. Yet, without an NFR parent of a child element can be associated with related FRs. An explicit specification of the NFRs association's agreements between the FRs is required. All those specification will be under the elements of Association. As an example of Association artifacts, the order is in place before the operation of the association reacts with faster response time to set high or low latency.

Then in the middle of the model, test cases are not associated only with the requirements but also with the NFRs. Based on the basic model of agile traceability models, the code will be traced back to the requirements based on test cases that associate with the requirements. Therefore, in this model, we also need to provide test cases that test NFRs without adding any code (depending on the kind NFRs). Thus, the impact of changes happen during the development will be traced by using test cases. This test cases will track back FR including NFRs.

Based on Figure 3, the black highlighted box is the part of NFR Traceability model and the blue highlighted box is of the Agile Traceability model. Therefore the overlap in blue and black highlighted box is the point where they are integrated. The integration validation is discussed in the next section.

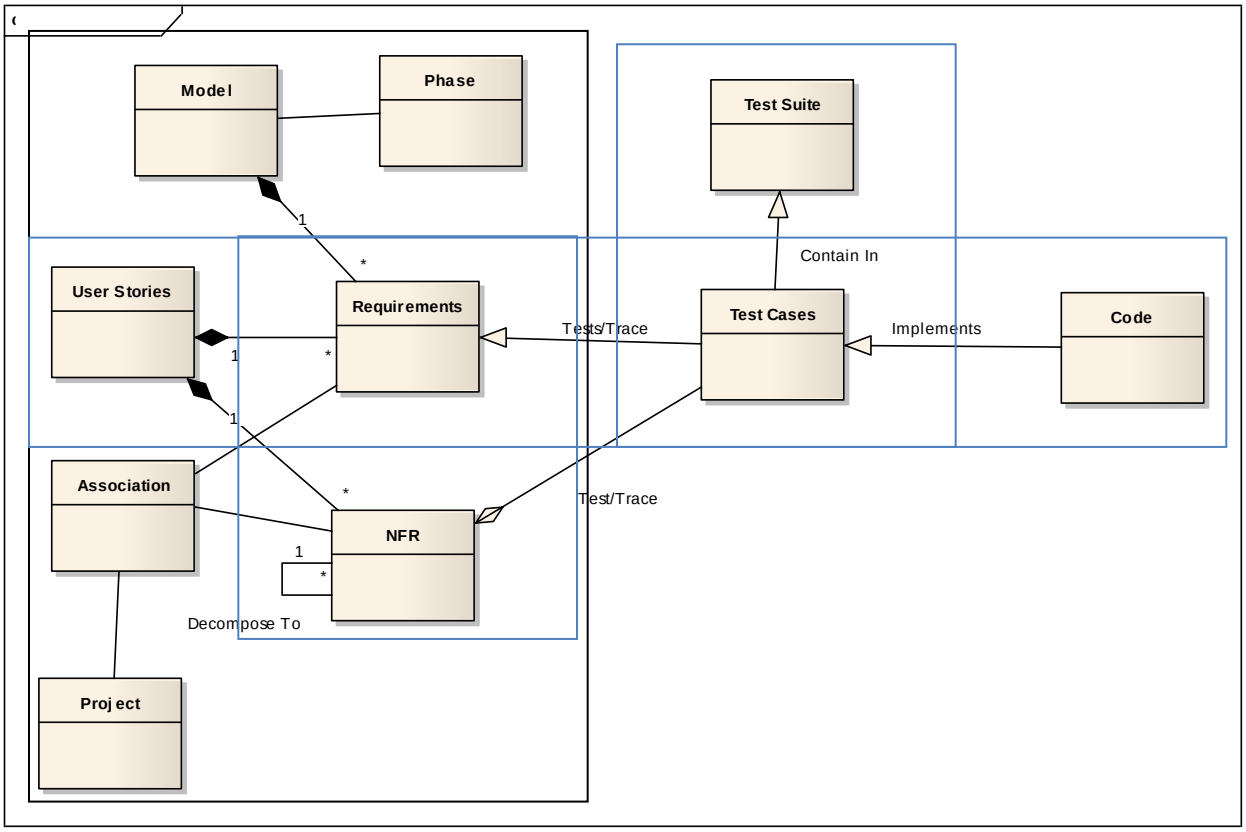


Figure 3 Proposed NFR Agile Traceability Model

5.0 INTERGRATION VALIDATION RESULT

The integration between ATM and NFRTM that form a new metamodel needs to be validated. This validation result determines whether the integration is compatible or not. In this validation phase, Precision and Recall method [22] are used. Precision has a mutual relationship with Recall, in which one thing affects or depends upon another.

In order to validate, ATM is declared as set A element that formed AATM set while set B element is NFRTM set that formed as BNFRTM. Based on Figure 1 and Figure 2, AATM consists of six elements whereas BNFRTM consists of seven elements. Based on Figure 3, the integration result shows that there are three elements that overlap. These elements are Requirements, FR and NFR. This technique subsequently uses a primary measurement ABmeasure. The equation for calculating ABmeasure utilize both Precision and Recall value to encounter any problems of misestimating of measurement. The equation to get ABmeasure will be shown below.

Basically, ABmeasure used four notions that are true positive (tp), true negative (tn), false positive (fp) and false negative (fn). However, in this calculation only three notions will be used. First, tp is the elements that overlap between AATM set and BNFRTM set that become a new set of integration elements set, ABintergrate set. Notion fp is used as the elements of AATM that do not overlap with BNFRTM set. While fn is the elements of BNFRTM that do not overlap with AATM. The calculation and formula for Precision and Recall are as below:

$$tp = |ABIntergrate| = 3 \quad \text{(Eq. 1)}$$

$$fp = |AATM| - |tp| = 6 - 3 = 3 \quad \text{(Eq. 2)}$$

$$fn = |BNFRTM| - |tp| = 7-3 = 4 \quad \text{(Eq. 3)}$$

The tp has 3 elements that are Requirement, FR and NFR. While fp consists of 3 elements that are Tests Suite, Test Case and Code and fn consists of four elements that are Association, Model, Project and Phase. Based on these values, Precision and recall could be count;

$$precision = |tp| / AATM = 3/6 = 0.5 \quad \text{(Eq. 4)}$$

$$recall = |tp| / BNFRTM = 3/7 = 0.43 \quad \text{(Eq. 5)}$$

From precision value and recall value, ABmeasure is calculated by this equation as show below.

$$ABmeasure = 2 \times (|tp| / (|fn| + |tp|) + (|tp| + |fn|) = 2 \times (3/13) = 0.46 \quad \text{(Eq. 6)}$$

Consequently, the integrated component model comprises a balanced average result value, where

ABmeasure is 0.46.This result is equally distributed with the average value of precision and recall.

Average Precision and Recall =
(Precision + Recall) / 2 = 0.5+ 0.43 /2 =0.4646 (Eq. 7)

The average value of precision and recall is 0.46. Therefore it could be concluded that the integration is mapped correctly.

6.0 TOOL SUPPORT

A tool called SAgile is developed to support this metamodel. Figure 4 shows the use case diagram used to develop the SAgile tool. There are four main actors: Product Owner (PO), Tester, Security Master and Developer. Based on the figure, the term "Manage" in a few use cases such as Manage Project and Manage User Stories refers to the role that is related to use cases for add, delete and edit function in each SAgile related feature. For example, based on the figure, PO role is connected with Manage Project use case. It means that the role of PO has the authority to add, delete and edit the project information using SAgile tool. SAgile is designed to help the PO, Security Master, Development team, and tester in managing the software development project in Agile manner. It also helps the development team to trace NFRs such as security and performance features in the system. Each of the metamodel component in this tool is discussed in the next section.

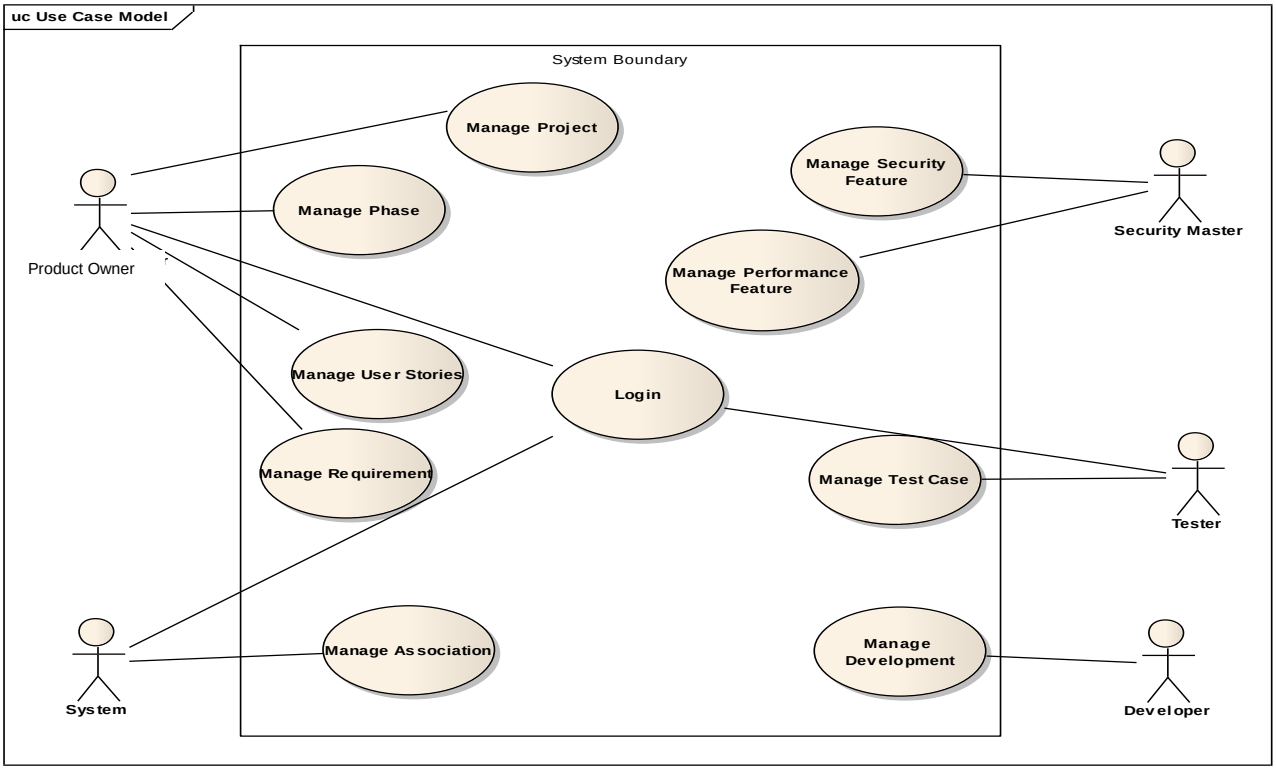


Figure 4 Use case diagram for SAgile tool

7.0 MAPPING OF METAMODEL TO TOOLS

In this section, we will show the mapping of Agile NFR Traceability Metamodel to tools that we develop known as SAgile. Figure 5 and Figure 6 show the features inside SAgile tool, representing each component in the Agile NFR traceability metamodel. SAgile tool has been improved from the previous work proposed in FDD [6], [23], [24], Scrum [25], [26] and XP [27] mainly with security features.

In order to show the mapping between Agile NFR Traceability Metamodel to tools, an experiment has been carried out by using SAgile in a small software development project that is called Hotel Management System. Basically, this system has a few user stories that we set and a few security and performance features that are linked to the user stories. Some of the user stories are linked with security features such as SQL injection or Cross Site Scripting (XSS), some are linked with both NFR security and

performance feature and some user stories are not linked to any of the NFR features. This could shows the difference indications that we have set in SAgile. Even though this is a controlled experiment, we tried to make this project having the same condition with a real software development project.

Before entering all the features, the Project Manager (for FDD projects) or Scrum Master (for Scrum project) must set the project that is called Hotel Management System belongs to Project element. After that, the Project Manager must set the iteration (for FDD project) or backlog (for Scrum project). This SAgile feature is under the element of Phase. Then, the Project Manager will pick the developer team, tester team or perhaps design team if needed. Then, they have to list out all the user stories. The list of user stories in one project belongs under Model element. Then the steps on how to link the user stories to NFR features will be explained next.

First we look at Figure 5 where there is a list of user stories. The first user story is in green color and the second user story is in blue color. The green color user story indicates that this user story is linked to any security and performance features while the blue colored user story shows that it does not link to any security or performance feature. This different color indicates to the development team whether a certain user story is linked to NFRs or not. This feature of SAgile belongs to the requirements, NFRs and user stories elements in our proposed metamodel.

Figure 5 User stories

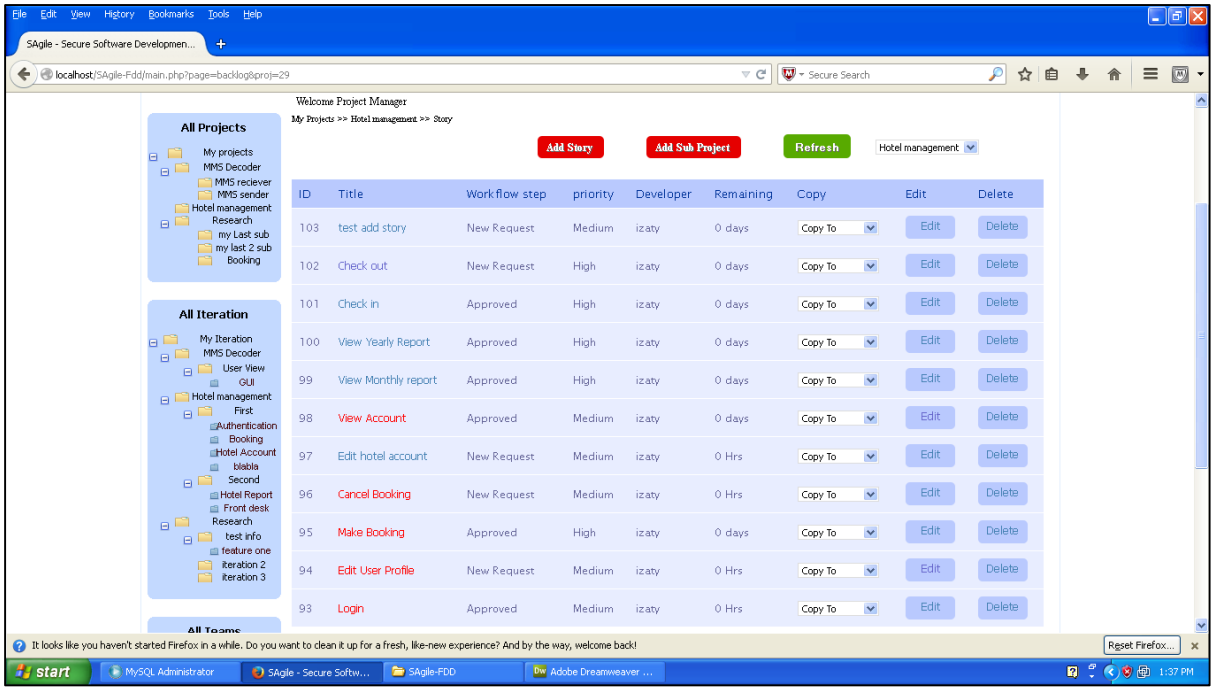


Figure 6 User stories (Security Feature)

Meanwhile in Figure 6, some of the user stories are in red color. This actually indicates that the feature actually is linked to security features only. Each colour code means diferent linkage of user store(functional requirement) with NFR. It mean that this feature is directly associated with "Manage Assocition" use case/ association element (refer to Figure 4).

Based on these two figures, we can see three categories of association element. The first category of association element is user stories or requirements elements that do not have linkage with any NFR features. The second category is user stories or requirements element that have one one linkage with one NFR element so in this case, it is one to many relationship. The third category is user stories or requirements element that have linkage with two or more(depends on the project need of tracking how many NFR types) NFR element, so that is one to many relationship. This give the flexibility for the devloper to link as many NFR needed or none at all if not required.

Then Figure 7 shows the example of performance and security features that are linked to the user stories. The checked box gives the development team to choose any NFR features that should be linked to the user stories. While this feature is added by the person in charge of taking care of the system quality or NFR. For example, we have SQL INJECTION in security Feature column and LOADING TIME in 'Performance Feature' column. All these are added before we linked them to each user stories. Hence, these features are linked to "Manage Security" & "Manage Performance" use cases.

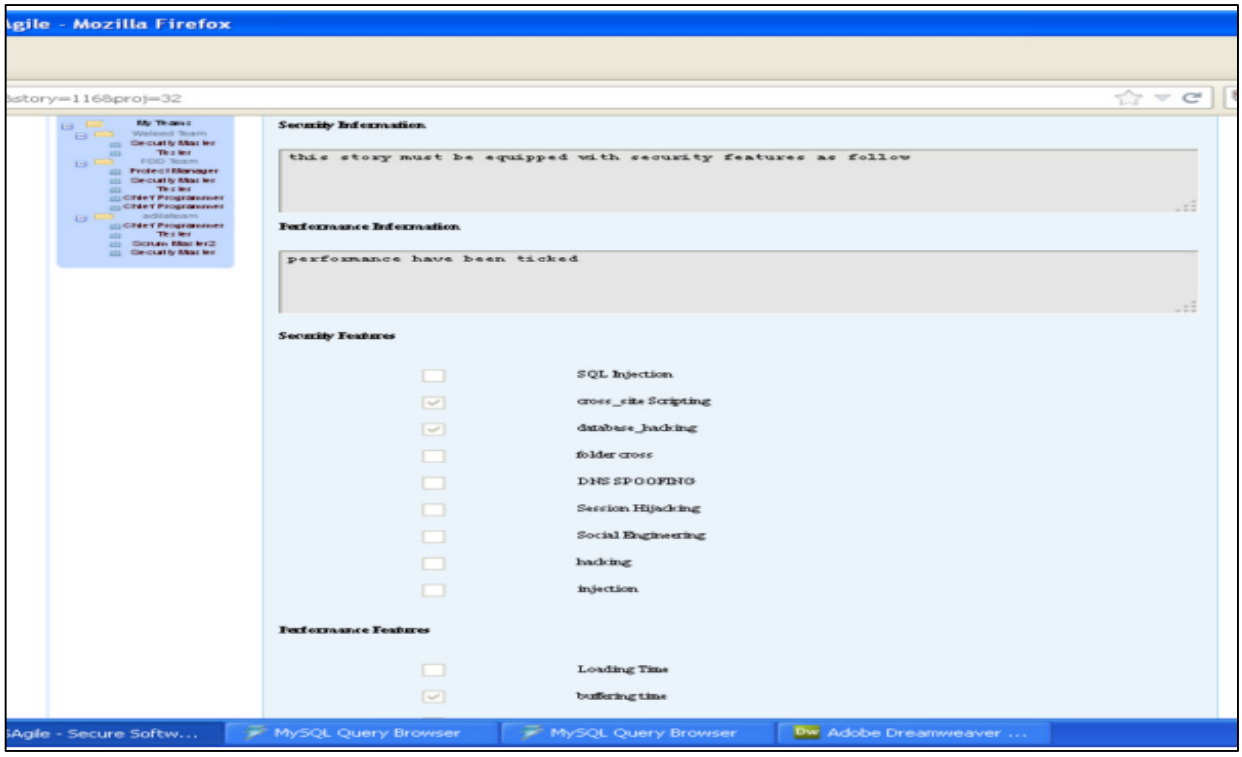


Figure 7 NFRs(Security & Performance)

Then "Manage Association" use case functions are not traced only in one way such as: user stories to NFR, or security to performance but also the linkage goes backward too. Hence based on Figure 8, the tool actually displays any user stories that have linked to any NFR. Also, the tool provides status information so that the team could know the progress of each user stories together with NFR.

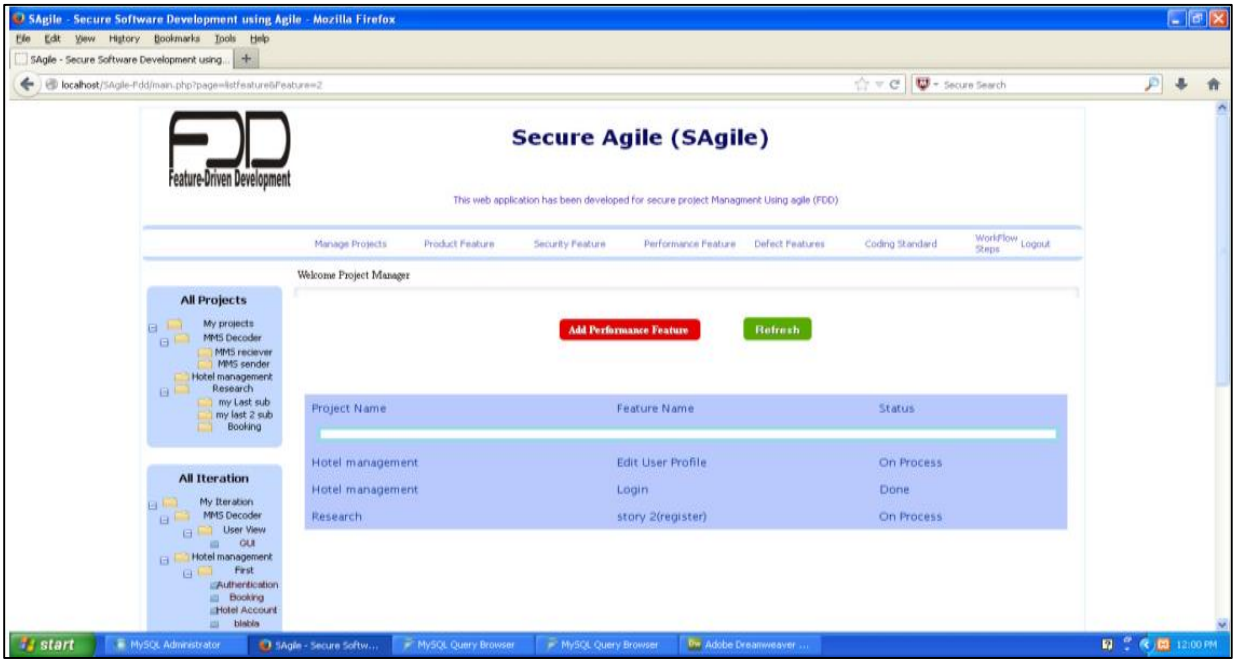


Figure 8 Association between User stories to NFR

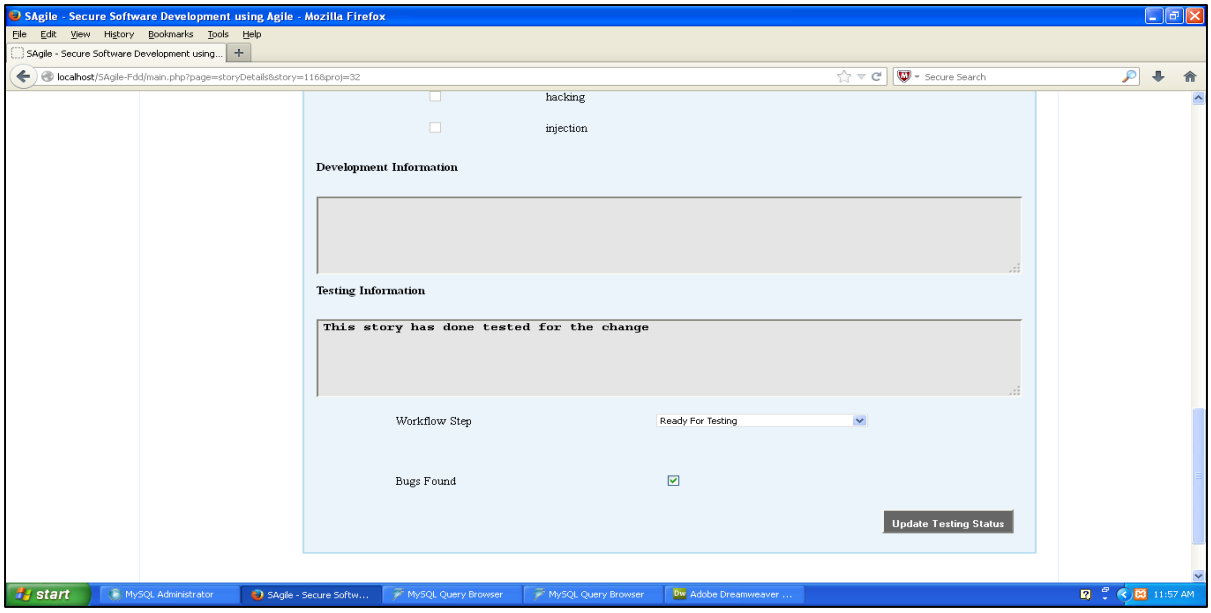


Figure 9 Association between User stories to Test Cases

Lastly, Figure 9 shows the link or association between each user story and test cases. This is important because basic Agile traceability concept suggests that the trace is not required from design to architecture. A simple link from user stories to test cases should complete the task. However, these test cases are not only limited for user stories but must also be eqquipped with NFR test cases. For example, test case one is linked to login user story that requires Sql injection mitigatiion so the test case case needs to be checked if the login works properly but also it must check that any attempt to to do sql injection toward the login should not be successful. Therefore, this function is related to "Manage Test" use case.

Then, one test case is mapped through the element of test case element but a full test suite element is the list of test case for each user stories individually, test case for each linkage of user stories to NFR features and test case for each NFR features individually. Finally, all the elements of Agile NFR Traceability Metamodel has been mapped back to Sagile tools feature. Therefore it is proven that Sagile tool are develop based on or guided by the Agile NFR Traceability Metamodel.

8.0 CONCLUSION AND FUTURE WORK

The main objective of this paper is to design and evaluated a metamodel that will be the guidance and benchmark for making an approach in helping the Agile development team to trace NFRs during Agile software development. This paper shows the integration between the Agile traceability model and Systematics NFR Traceability Model. This paper also shows the flow of integration of these two metamodels that produces a new integrated metamodel and validate the integrated metamodel using Precision and Recall method. Therefore the integrated metamodel could be used as a benchmark in producing an approach that solves the issues of tracing NFR such as security and performance features during Agile software development.

In addition, the integration also offers flexibility during modelling of components. This offers more specific modelling artefacts and more details of design models, compared with a model using UML 2.0. This integrated metamodel has the potential to be a guideline in designing traceability approach in Agile software development. The authors suggest that the future work needs to be done continuously since there are still some issues as discussed in earlier section in this paper. Future work includes the study of other NFRs with respect to traceability.

Acknowledgement

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] Breivold, H. P, D. Sundmark, P. Wallin, and S. Larsson 2010. What Does Research Say about Agile and Architecture. *Fifth International Conference on Software Engineering Advances* 32-37. Retrieved April 1, 2013 (http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5615620).

[2] Abbors, F., J. Lilius, A Joukahaisenkatu, E. Fredrik Abbors, and D. Truscan. 2009. Tracing Requirements In A Model-Based Testing Approach. *First International Conference on Advances in System Testing and Validation Lifecycle*. 123-128.

[3] Chung, L., B. A. Nixon. & Yu, E. 1996. Dealing with Change : An Approach Using Non-functional Requirements. *Requirements Eng*. 1: 238-260, 238-260.

[4] Grau, R. 2012. *Requirements Engineering in Agile Software Development. Software for People*. Springer Berlin Heidelberg

[5] Huang, J. C., A. Czauderna, and E. Keenan. 2013. *A Persona-Based Approach for Exploring Architecturally Significant Requirements in Agile Projects*. Requirements Engineering: Foundation for Software Quality. Springer Berlin Heidelberg.

[6] Firdaus, A., Ghani, I, and Jeong, S. R. 2014. Secure Feature Driven Development (SFDD) Model for Secure Software Development. *Procedia-Social and Behavioral Sciences.* 129: 546-553.

[7] Cao, L., Mohan, K., Ramesh, B., & Sarkar, S. 2013. Adapting Funding Processes for Agile IT Projects: An Empirical Investigation. *European Journal of Information Systems.* 22(2): 191-205.

[8] VanderLeest, S., and A. Buter. 2009. Escape the Waterfall: Agile for Aerospace *28th Digital Avionics Systems Conference.*

[9] Grundy, J., J. Hosking, W. B. Mugridge, 2002. Inconsistency Management for Multiple-View Software Development Environments. *Software Engineering, IEEE Transactions on.* 24(11).

[10] Gotel, O., C., Z., and Finkelstein, C.W, 1994. *An analysis of the requirements traceability problem*. In: Proceedings of the First International Conference on Requirements Engineering, pp. 94–101.

[11] Reshef, A. N., R. F. Paige, J. Rubin, Y. Shaham-Gafni, and D. S. Kolovos.2005. *Operational Semantics for Traceability* In Proceedings of ECMDA Traceability Workshop, ECMDA-TW '05, pages 8–14

[12] Ajila, S., and A. B. Kaba. 2004. Using Traceability Mechanisms to Support Software Product Line Evolution. In *Information Reuse and Integration, 2004. IRI 2004. Proceedings of the 2004 IEEE International Conference on IEEE.* 157-162.

[13] Chen, J. Y., & S. C. Chou. 1999. Consistency Management in a Process Environment. *Journal of Systems and Software.* 47(2): 105-110.

[14] Huang, J. C. 2005. Toward Improved Traceability of Non-Functional Requirements. *Proceedings of the 3rd International Workshop on Traceability in Emerging Forms of Software Engineering. ACM.*

[15] Kassab, M., and O. Ormandjieva. 2006. *Towards an Aspect-Oriented Software Development Model with Tractability Mechanism.* Trese.cs.utwente.nl. 1–8.

[16] A. G. Kourie and B. W. Watson. 2003. Standards and Agile Software Development. Proceedings of SAICSIT 2003. 1-11.

[17] Huang, J. C., O. Gotel, and A. Zisman. 2012. *Software and Systems Traceability.* 2(3). Springer.

[18] Collins, E. F., and V. F. de Lucena. 2012. Software Test Automation Practices in Agile Development Environment: An Industry Experience Report. Automation of Software Test (AST). *7th International Workshop on. IEEE.*

[19] Julian, R., and J. Green. 2004. Automating Traceability For Generated Software Artifacts. *Proceedings of the 19th IEEE International Conference on Automated Software Engineering.* IEEE Computer Society.

[20] Kassab, M., O. Ormandjieva, and Maya Daneva. 2009. A Metamodel for Tracing Non-functional Requirements. *Computer Science and Information Engineering, 2009 WRI World Congress on.* 7. IEEE.

[21] Huang J. C., R. Settimi, O. BenKhadra, E. Berezhanskaya, and S. Christina. 2005. Goal Centric Traceability for Managing Non-Functional Requirements. *Proceedings of the 27th International Conference on Software Engineering.* 362-371

[22] Kappel, G., Kargl, H.,Kramler, G., Schauerhuber, A., Seidl, M. Strommer, M., & Wimmer, M. 2007. Matching Metamodel with Sematics Systems-An Experience Report. Datebanksysteme in Business. *Technologie und Web Workshop Proceedings, 5.-6 Marz, Aachen Germany*. 38-52.

[23] Firdaus A., Ghani I., and Yasin NIM. 2013. Developing Secure Websites Using Feature Driven Development (FDD): A Case Study. *Journal of Clean Energy Technologies Issue*. 1(4).

[24] Inayat, I., Salim, S. S., Marczak, S., Daneva, M., & Shamshirband, S. 2014. A Systematic Literature Review on Agile Requirements Engineering Practices and Challenges. *Computers in Human Behavior*.

[25] Azham Z., Ghani I., and Ithnin N. 2011. Security Backlog In Scrum Security Practices. Software Engineering (Mysec). *2011 5th Malaysian Conference in.* 414-417.

[26] Ghani I., Azham Z., and Jeong SR. 2014. Integrating Software Security into Agile-Scrum Method. *KSII Transactions on Internet and Information Systems (TIIS)*. 8(2): 646-663.

[27] Azham Z., Ghani I., and Ithnin N. 2011. Security Backlog In Scrum Security Practices. Software Engineering (Mysec). *2011 5th Malaysian Conference in*. 414-417.