2012 19th Asia-Pacific Software Engineering Conference

The NORMAP Methodology: Lightweight Engineering of Non-functional Requirements for Agile Processes

Weam M. Farid
Graduate School of Computer and Information Sciences
Nova Southeastern University
Fort Lauderdale, USA
weam@nova.edu

***Abstract*— Agile software development methodologies, such as Scrum, have become very popular in quickly delivering quality Functional Requirements (FRs). However, agile methodologies have not sufficiently identified, modeled, and linked Non-Functional Requirements (NFRs)—such as security and performance—with FRs in early requirements phases. This research presents a lightweight engineering of NFRs for agile processes. The proposed Non-functional Requirements Modeling for Agile Processes (NORMAP) Methodology identifies, links, and models Agile Loose Cases (ALCs) with Agile Use Cases (AUCs) and Agile Choose Cases (ACCs). A lightweight adapted version of the NFR Framework was developed including 25 important NFRs. Further, a risk-driven agile requirements implementation sequence and a visual tree-like view were developed. The methodology was validated through developing a Java-based modeling simulation tool and two case studies.**

NORMAP methodology; Scrum; agile requirements modeling; agile non-functional requirements; NORMATIC

I. INTRODUCTION

Agile software development methodologies have not adequately identified, modeled, and linked Non-Functional Requirements (NFRs) with Functional Requirements (FRs) during early development phases. Researchers have agreed that NFRs have been generally ignored, ill-defined, dealt with as an afterthought process, not treated as first-class artifacts during conventional Requirements Engineering phases, and especially ignored in agile methodologies [1][2][4]. Academic research and industrial case studies suggest that agile software development methodologies, such as Scrum and Extreme Programming (XP), have become popular in quickly delivering FRs [3]. However, agile approaches lack support for NFRs.

Agile methodologies, such as Scrum, have gained global momentum and are expected to grow in the next 10 to 15 years [5]. Scrum has been reported to incrementally deliver complex software solutions with improved quality in shorter time frames while preferring customer involvement over documentation and heavyweight processes. However, agile requirements are handled differently from traditional processes such as Waterfall [1]. Authors in [2][5] suggested that there is a significant lack of NFRs identification, modeling, and linking with FRs particularly in agile environments and that agile development methods are criticized for lacking explicit practices for NFRs. In addition, Scrum requires engineering excellence to accomplish inspection and adaptation [6]. Therefore, merging engineering with agile practices is a desirable objective. Agile processes ought to provide techniques to identify NFRs while user stories are being scoped with customers [7]. This paper presents a lightweight engineering yet agile methodology for identifying, linking, and modeling NFRs with FRs. The paper is organized as follows. Section 2 presents the methodology and sections 3 and 4 present two case studies. Section 5 provides a summary and future work.

II. NON-FUNCTIONAL REQUIREMENTS MODELING FOR AGILE PROCESSES (NORMAP) METHODOLOGY

A. Goals of the NORMAP Methodology

The objective of this study is to research and develop a simple yet systematic NFR modeling framework that is tailored for agile processes. The framework design objectives include combining both FRs and NFRs in one framework, treating NFRs as first-class artifacts and linking them to FRs, proposing the new W^8 Story Card Model that captures FRs, using a color-coded downscaled version of Chung's NFR Framework [8], proposing new requirements quality and project management metrics, developing a simulation tool, and computing and viewing a risk-based requirements implementation sequence.

B. Concepts in The NORMAP Framework

Three areas of enhancements were proposed to support the objectives. First, the W^8 User Story Card Model [14] is proposed as an enhancement of the agile index card technique used to gather requirements in XP and Scrum. The story card is a physical 3x5 index card used to capture 3-5 sentences of user requirements from business owners. On the other hand, the traditional use case model is heavyweight and stifles agility. A good balance is to blend the two approaches and capture only important elements of a requirement and other parameters.

The first "W" in the W^8 Model captures the "who" or the actor. The second "W" is the "what" or the desired action that the user wants. Next is the "why" or the business justification for the desired functionality. The fourth "W" is the "Without ignoring" which indicates linguistically significant terms that may map to a system quality or a required NFR (e.g. "Without ignoring security"). The fifth "W" captures the "While it is nice to have" or optional NFRs. The sixth "W" captures the "Within" or an initially estimated time frame to deliver the requirement. The seventh "W" captures "With a priority of" which is the business priority. The last "W" ("Which may impact") captures a list of impacted requirements.

Second, NORMAP proposed an Agile Requirement taxonomy which classified requirements as either

1530-1362/12 $26.00 © 2012 IEEE
DOI 10.1109/APSEC.2012.23

functional (Agile Use Case—AUC) or non-functional (Agile Loose Case—ALC) [14]. An Agile Choose Case (ACC) [14] is also part of the proposed taxonomy and represents potential solutions of NFRs according to Chung's NFR Framework [8]. A light-colored non-bold cloud represents an Agile Loose Case while a dark-colored bold cloud represents an Agile Choose Case (Figure I).

Third, NORMAP utilized the well-known Aspect-Oriented "pointcut" operators (*Before, After, Override, and Wrap*) in linking functional and non-functional requirements. The operators simply indicate a point at which an Aspect is "woven" into another piece of source code to eliminate code tangling and crosscutting concerns [9]. In NORMAP, a pointcut simply refers to a point in time at which an ACC is carried out and linked to an AUC. The pointcuts visually map to the four physical edges of a story card (depicted as a yellow sticky note as in Figure 1), such that the "Before" pointcut maps to the left edge, the "After" pointcut maps to the right edge, the "Override" pointcut maps to the top edge, and the "Wrap" pointcut maps to the bottom edge [14]. If a potential ACC needs to be executed before or after an AUC, then the ACC must be linked to the AUC's left or right edges, respectively. If an ACC overrides an entire AUC, then the top edge of the card is used. If an ACC is executed before and after an AUC, then the *Wrap* operator (bottom edge) is used.

C. The NORMAP Methodology

NORMAP is a lightweight conceptual framework for modeling NFRs in agile processes. The framework can be used manually or semi-automatically. The manual framework is called Non-functional Requirements Modeling for Agile Manual (NORMANUAL) practices. The Non-functional Requirements Modeling with Agile Automatic (NORMATIC) tool [15] supports semi-automatic implementation of the framework. The framework is divided into layers that support Agile Use Cases, NFRs meta-database (NORBASE), Agile Loose Cases (NFRs), Agile Choose Cases (potential solutions), Non-functional Requirements Plan (NORPLAN), and a tree view of this plan (NORVIEW). Table I shows seven development phases of NORMAP and NORMATIC.

TABLE I. NORMAP'S MAJOR DEVELOPMENT PHASES

ID	*Description*
1	*Selected NFRs and Initial Data Collection*: The NFR Framework [8] has 161 high-level NFRs. Only 25 selected NFRs that satisfied certain criteria were used in this study.
2	*Initial Data Pre-Processing*: WordNet [10], a natural language processing tool, was used to derive synonyms and antonyms of the 25 selected NFRs and store it in the database (NORBASE).
3	*Automatic Parsing of Requirement Statements*: Stanford Parser [11] was used to parse sentences and identify verbs, subjects, adjectives, and adverbs, and populate the W^8 Story Card Model.
4	*Modeling Agile Use Cases*: In NORMAP, an Agile Use Case (AUC) combines a simplified version of a traditional use case with an expanded version of an agile user story [14]. An AUC is modeled as a yellow sticky note surrounded by 4 pointcuts.
5	*Modeling Agile Loose Cases*: In NORMAP, an Agile Loose Case (ALC) models a specific instance of an NFR [14]. An ALC is linked to one or more Agile Choose Cases (ACCs)—or potential solutions. One ALC may impact other ALCs. An ALC includes project management metrics and other attributes.
6	*Modeling Agile Choose Cases*: In NORMAP, an Agile Choose Case (ACC) represents one or more instances of potential solutions for the identified ALC [14]. An ACC captures basic project management metrics similar to ALCs. If an ALC has more than one ACC, visual Boolean operators ("AND" and "OR") must link solutions to the parent ALC.
7	*Requirements Implementation Sequence Planning*: The framework supports agile non-functional requirements planning (NORPLAN) using project management and requirements quality metrics to compute a risk-driven implementation sequence, which is viewed as a tree (NORVIEW).

III. CASE STUDY WITH THE PROMISE DATASET

A. Background

The first case study used to validate the NORMAP Methodology utilized the Predictor Models in Software Engineering (PROMISE) dataset used in NFRs classification [12]. The dataset represented a set of unrelated requirements from 15 different projects in a software engineering course at DePaul University. The case study was used to evaluate the NORMAP Methodology's ability to parse independent requirements (through a batch process) and identify possible Agile Loose Cases (NFRs). After refining the dataset and eliminating requirements that were not part of the selected 25 NFRs for this study, a total of 607 out of the original 635 requirements statements were used from the dataset.

B. Results

Results obtained from parsing and classifying NFRs in the PROMISE dataset were categorized as *Success*, *Partial Success*, or *Failure*. A user requirement was classified as *Success* if NORMAP correctly identified the same NFR type in the original PROMISE dataset. If NORMAP could not identify any NFRs for a pure functional requirement, then it was classified as a *Success*. If NORMAP "closely" identified at least one NFR, then the requirement's classification was considered *Partial Success*. If NORMAP could not identify a predetermined NFR, then it was classified as *Failure*. A special batch process was developed without any visual modeling. Table II shows a summary of the parsing and classification results obtained in this case study with an overall success rate of 87.15%.

TABLE II. PROMISE DATASET PARSING/CLASSIFICATION RESULTS

Requirement		**Detailed Results**					
Type	***Count***	***Failure***		***Partial Success***		***Success***	
		Count	***%***	***Count***	***%***	***Count***	***%***
F	252	0	0.00	241	39.7	11	1.81
U	67	12	1.97	0	0.00	55	9.06
S	66	18	2.96	0	0.00	48	7.90
O	62	12	1.97	0	0.00	50	8.23
P	53	1	0.16	0	0.00	52	8.56
LF	37	9	1.48	0	0.00	28	4.61
SC	21	12	1.97	0	0.00	9	1.48
AV	21	0	0.00	0	0.00	21	3.45
M	15	4	0.65	0	0.00	11	1.81
LG	13	10	1.64	0	0.00	3	0.49
Total	**607**	**78**	**12.8**	**241**	**39.7**	**288**	**47.4**
Overall Results: 529 out of 607 (87.15%) Success Rate							
Legend: F=Functional; U=Usability; S=Security; O=Operational; P=Performance; LF=Look and Feel; SC=Scalability; AV=Availability; M=Maintainability; LG=Legal							

C. Analysis

NORMATIC used the Stanford Parser to parse requirement statements into the first four elements of the W[8] User Story Model. The actor was identified through the nominal subject. The "without ignoring" data element was populated by identifying similar nouns or adjectives related to significant words parsed from the sentence. In some instances, the derived nouns or adjectives were not grammatically valid in conjunction with the sentence. However, it was important to flag significant words in the sentence that might help identify possible NFRs even if the lexical form of nouns or adjectives might be inconsistent with the fidelity of the sentence.

To illustrate, a requirement statement, which reads "*The system shall refresh the display every 60 seconds.*", was parsed and the actor was correctly identified as "*system*", and the action was identified as "*refresh the display every 60 seconds*". Furthermore, since no justification was present in the sentence, the third element of the W[8] User Story Card Model was left blank. In addition, NORMATIC identified "seconds" as a significant term that matched a keyword related to "Performance" NFR (through a related "Time" concept). Therefore, NORMATIC successfully identified "Performance" as a possible NFR.

IV. Case Study with the Eu Procurement System

A. Background

The second case study utilized the requirements model of the European Union (EU) eProcurement Online System [13], which included 26 functional requirement statements from which other NFRs were identified using the NORMAP Methodology. This validation focused on visually identifying and linking AUCs and ALCs and computing and visually modeling a risk-driven requirements implementation sequence.

B. Modeling Steps

Table III summarizes the modeling steps that were carried out in this case study using NORMATIC.

TABLE III. Agile Requirements Modeling Steps

Step	Description
1	Enter each of the 26 FRs as an Agile Use Case using NORMATIC. For each AUC, the entire original description of each requirement statement was entered and NORMATIC parsed the full story (only 1 sentence as the main user story).
2	Identify and classify possible Agile Loose Cases for each Agile Use Case using Natural Language Processing tools.
3	Link/weave correctly identified ALCs to their corresponding AUCs through one of the four pointcuts (*Before, After, Override, Wrap*) depending on the requirement's context.
4	Link ALCs to Agile Choose Case(s) (potential solutions).
5	Update project management and requirements quality metrics and their impacts such as Sprint Duration, Estimate Scheme, Team Development Factor, Team Velocity, Agile Process Maturity, Requirements Ambiguity, Completeness, and others
6	Enter relative size estimates for requirements and potential solutions using Planning Poker cards technique. A final PERT estimate is calculated as follows: PERT Estimate = (P+4M+O)/6 [P=Pessimistic, M=Most-Likely, O=Optimistic].
7	NORMATIC calculates/updates risk scores of requirements.
8	Conduct three experiments for each Priority Scheme (Table IV) to generate/visualize the implementation sequence plan.

C. Modeling Example

The User Profiling requirement (Figure 1) included three ALCs: Auditability, User Interface, and Usability. Since Auditability required storing a transaction trail of every user action performed, a potential source code-based Aspect can be woven into the AUC's *After* pointcut to capture user identity, actions performed, date/time after the functionality is executed. Therefore, the Auditability ALC and the Auditing Aspect ACC were modeled as light and dark green clouds, respectively. For the User Interface ALC, the requirement stated that users should be able to select and save their user interface preferences. As a result, Configurable UI was identified as a potential architectural ACC (dark red cloud) that was linked to its associated ALC (light red cloud), which in turn was linked to the AUC via the *Wrap* pointcut. The Usability ALC (light blue cloud) was also identified and addressed by an organizational policy solution, such as online training (dark blue cloud). Further, addressing the User Interface ALC could also "strongly help" the Usability ALC, which was modeled by the "++(85%)" labeled on the edge connecting the two ALCs. A green checkmark on the edge between an ALC and an ACC validates that the potential solution matches the type of NFR.

D. Requirements Implementation Sequence

After all requirements quality and project management metrics were entered into NORMATIC, the model was stored as a GraphML (XML) file. The Requirements Implementation Sequence (NORPLAN) along with a tree-like visual model (NORVIEW) were computed and generated. The total relative size estimate for all requirements was 479 User Story Points (USPs). Sprint duration was set to 2 weeks and Team Velocity was set to 30 USPs. The validation process included running three experiments with three priority schemes (Table IV).

E. Results and Analysis

This case study included 26 FRs and over 100 requirement sentences of which 57 included possible NFRs. NORMATIC was successful in fully or partially parsing and classifying 50 out of 57 user requirements sentences that contained NFRs—an overall success rate of 87.71% with 16 identified ALCs.

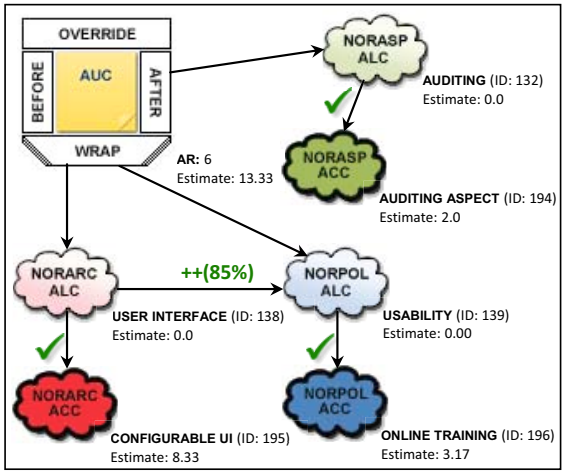

Figure 1. Agile Modeling of the User Profiling Requirement

Results of Experiment 1 suggested that using Scrum's business value priority scheme, implementation would take 6 releases (21 sprints) starting on 1/3/2011 and ending on 11/14/2011. This prioritization scheme does not account for any risk factors. Results of Experiment 2 showed that implementing the riskiest requirements first shortened the duration by two months and required only 5 releases (17 sprints). Sprints 1 and 2 addressed only NFRs such as Confidentiality, Security, User Interface, and Documentation, as early as possible so that they can be reused by subsequent AUCs. This experiment supported agile best practices that promote "failing quick, early, and small" rather than "failing slow, late, and too big". Results of Experiment 3 showed that when the riskiest requirements were implemented last, the implementation duration was shortened by only one month and required 5 releases (19 sprints). This result was in line with the notion that risk becomes more expensive to mitigate later than if addressed earlier. Figure 2 shows the Burndown Chart of 479 User Story Points across all releases. The most agile plan was achieved using Riskiest-Requirements-First priority scheme (17 sprints) followed by the Riskiest-Requirements-Last (19 sprints). The highest business value priority scheme ranked third (21 sprints).

V. Conclusion and Future Work

This research study proposed and investigated the NORMAP Methodology, a lightweight engineering framework for identifying, modeling, and planning NFRs and linking them with functional requirements in agile processes. The framework utilized a risk-driven approach in calculating and visualizing an improved requirements implementation sequence based upon newly proposed requirements quality and project management metrics. A visual modeling simulation tool, NORMATIC, was designed and developed using visualization frameworks and Natural Language Processing tools for NFRs identification. Two case studies validated the NORMAP Methodology with promising results. The study showed that a quantifiable and risk-driven prioritization scheme resulted in a systematic yet more agile plan. The promising results call for further research to validate the framework in real-world agile development teams, improve NFRs identification and text mining, and integrate the framework with web, mobile, and social media technologies.

TABLE IV. Summary of Three Validation Experiments

ID	***Experiment Description***
1	*Highest Business Value Implementation Sequence*: The requirements implementation sequence was computed based upon the highest business value without considering any technical or project management risks. The order of listing requirements in the specifications simulated business value.
2	*Riskiest-Requirements-First Implementation Sequence*: Risk-based algorithm computed the sequence using riskiest requirements first. Risks were calculated based upon requirements quality and basic project management metrics.
3	*Riskiest-Requirements-Last Implementation Sequence*: The same risk-based algorithm computed the requirements implementation sequence using riskiest requirements last.

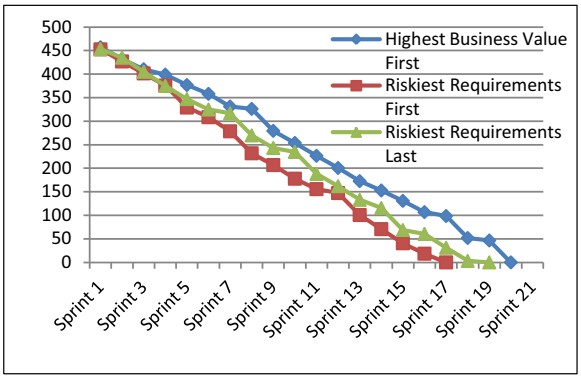

Figure 2. Burndown Chart of User Story Points By Priority Scheme

References

[1] N. Mead, V. Viswanathan, and D. Padmanabhan, "Requirements Engineering into the Dynamic Systems Development Method", Proc. of the 2008 32nd Annual IEEE International Computer Software and Applications Conference*, 2008,* pp. 949-954.

[2] J. Araujo and J. Ribeiro, "Towards an Aspect-Oriented Agile Requirements Approach", In proceedings of the Eighth International Workshop on Principles of Software Evolution (IWPSE'05), 2005, pp. 140-143.

[3] A. Marcal, F. Furtado Soares, and A. Belchior, "Mapping CMMI Project Management Process Areas to SCRUM Practices", In 31st IEEE Software Engineering Workshop (SEW 2007), 2007.

[4] F. Paetsch, A. Eberlein, and F. Maurer, "Requirements Engineering and Agile Software Development", In IEEE Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003, pp. 308.

[5] C. Ho, M. Johnson, L. Williams, and E.M. Maximilien, "On Agile Performance Requirements Specification and Testing", Proc. of the AGILE Conference (AGILE'06), 2006, pp. 47-52.

[6] K. Schwaber, Agile Project Management with Scrum, Redmond, Washington, Microsoft Press, 2004.

[7] A. Eberlein and J. Leite, "Agile Requirements Definition: A View from Requirements Engineering", In Proceedings of the International Workshop on Time-Constrained Requirements Engineering (TCRE'02), 2002.

[8] L. Chung, B.A. Nixon, E. Yu, and J. Mylopoulos, Non-functional Requirements in Software Engineering, Boston, MA, Kluwer Academic Publisher, 2000.

[9] G. Sousa, G. Silva, and J. Castro, "Adapting the NFR Framework to Aspect- Oriented Requirements Engineering", In The 17th Brazilian Symposium on Software Engineering (SBES), 2003, Manuas, Brazil.

[10] WordNet, "WordNet, A Lexical Database for English", Internet: http://wordnet.princeton.edu, [Oct. 4, 2009].

[11] Stanford Parser, "The Stanford Parser: A Statistical Parser", Internet: http://wordnet.princeton.edu, [Oct. 4, 2009].

[12] G. Boetticher, T. Menzies, and T. Ostrand, "PROMISE Repository of Empirical Software Engineering Data", West Virginia University, Dept. of Computer Science, Internet: http://promisedata.org/?p=38, 2007 [May 17, 2010].

[13] European Dynamics S.A., "Functional Requirements for Conducting Electronic Public Procurement Under the European Union (EU) Framework Volume I", Internet: http://ec.europa.eu/internal_market/publicprocurement/docs/eprocurement/functional-reguirements-vol1_en.pdf, 2005 [Oct. 13, 2009].

[14] W.M. Farid and F.J. Mitropoulos, "Novel lightweight engineering artifacts for modeling non-functional requirements in agile processes", Proc. IEEE SoutheastCon 2012 (SoutheastCon 2012), Mar. 2012.

[15] W.M. Farid and F.J. Mitropoulos, "NORMATIC: A visual tool for modeling non-functional requirements in agile processes", Proc. IEEE SoutheastCon 2012 (SoutheastCon 2012), Mar. 2012.