NORMATIC: A Visual Tool for Modeling Non-functional Requirements in Agile Processes

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

Abstract— **There is consensus in the research community that agile software development methodologies, such as Scrum, are becoming more and more popular in delivering quality Functional Requirements (FRs). However, agile methodologies have not adequately modeled Non-Functional Requirements (NFRs) and their potential solutions (operationalizations) with FRs in early development phases—let alone having tools to support such initiatives. This research proposes NORMATIC, a Java-based simulation tool for modeling non-functional requirements for semi-automatic agile processes. NORMATIC is the semi-automatic tool that supports the more general Non-Functional Requirements Modeling for Agile Processes (NORMAP) Methodology. Early results show that the tool can potentially help agile software development teams in reasoning about and visually modeling NFRs as first-class artifacts early on during requirements gathering and analysis phases. The tool can also aid project managers and Scrum teams in user story estimate and risk calculations as well as risk-driven planning and visualization of the proposed plans.**

Agile Requirements Modeling; Agile Use Case; Agile Loose Case; Agile Choose Case; NFRs; Scrum; NORMATIC; NORMAP Methodology

I. Introduction

Agile software development methodologies, such as Scrum and Extreme Programming (XP), have been gaining traction and adoption on a global scale and are expected to grow in the next 10 to 15 years [1]. However, software engineers are constantly faced with challenges to deliver quality software under tight schedules, hence, pressuring developers to quickly start coding with “ad hoc short cuts” to accelerate the development phase [2]. Therefore, such ad-hoc methods suffer from the lack of a well-defined structure and tool support.

More specifically, agile software development methodologies have not adequately identified, modeled, and linked Non-Functional Requirements (NFRs) and their potential solutions (operationalizations) with Functional Requirements (FRs) during early requirements analysis phases. Researchers agree that NFRs have been traditionally ill-defined during conventional requirements engineering phases and especially ignored in agile methodologies [3][4].

Academic research and industrial case studies suggest that agile software development methodologies, such as Scrum and Extreme Programming (XP), have been gaining tremendous popularity and proven to be successful in implementing and quickly delivering quality FRs [5][6]. However, non-functional requirements (and often crosscutting concerns such as security, performance, and scalability) have been traditionally ignored or at best ill-defined in agile environments [7]. This research study is aimed at introducing the Non-functional Requirements Modeling for Agile Automatic (NORMATIC) processes—a Java simulation tool that supports the more general Non-functional Requirements Modeling for Agile Processes (NORMAP) Methodology.

This paper presents how NORMATIC supports the three building blocks of the lightweight engineering-based yet agile NORMAP methodology for identifying, linking, and modeling NFRs with FRs in an attempt to potentially improve software quality without compromising agility. The paper is organized as follows. Section 2 presents a brief literature survey and objectives of the tool. Section 3 describes the details of NORMATIC and how it links to its underlying framework. Section 4 summarizes initial results obtained using the tool to reason about and visually modeling NFRs in a simulated Scrum environment using a case study. Section 5 discusses the findings and presents areas for future work.

II. Literature Survey and Objectives

A. *Lack of Support for NFRs in Agile Processes*

Authors in [6] clearly stated that there was a significant lack of NFRs identification, modeling, and linking with FRs particularly in agile environments. Another research effort in [8] that supported this claim suggested that agile development methods are sometimes criticized for not having explicit practices for NFRs. Authors in [8] proposed the Performance Requirements Evolution Model (PREM) but it focused more on the specification and testing of one NFR—namely Performance—as opposed to a more generic NFRs framework.

Research by [9] also confirmed the semantic gap that exists between performance crosscutting concerns and functional concerns, leading developers to ignore the issue altogether. Many researchers claimed that NFRs were typically dealt with as an "afterthought" process and not treated as first-class artifacts during the software requirements phase [5]. Because agile processes such as Scrum requires engineering excellence to accomplish inspection and adaptation [10], it is necessary to develop new tools that support lightweight engineering yet agile processes to help requirements engineers in such agile environments. An unquestionable need for engineering of non-functional requirements in agile processes was well articulated when [11] suggested that agile processes ought to include explicit techniques to identify NFRs early on during requirements analysis—ideally while user stories are being scoped with stakeholders.

B. Objectives

According to [12], defined processes are vital elements in software process improvement but in order to realize the benefits, the model should simplify the complexity of the real world. The fundamental objective of NORMATIC is to develop a simple non-functional requirements modeling tool that specifically supports agile software development processes such as Scrum. NORMATIC implements the three building blocks of the more general NORMAP framework: Agile Use Case (AUC), Agile Loose Case (ALC), and Agile Choose Case (ACC). NORMATIC also provides support for planning agile releases and iterations (sprints) and visualizing such plans in a tree-like structure. NORMATIC's objectives are as follows:

- Integrate FRs and NFRs modeling under one agile tool.
- Parse user stories automatically to improve agility.
- Identify and classify NFRs to improve agility.
- Support a newly proposed and enhanced W^8 Story Card Model that captures functional or non-functional agile requirements.
- Visually classify NFRs and their potential solutions into three types: source code, architecture and design, and organizational policies.
- Improve visualization through a downscaled version of Chung's NFR Framework [13] with new color-coding of clouds that is suitable for agile processes to model NFR soft goals and their potential solutions.
- Provide extensibility through configurable project management and requirements quality metrics.
- Implement a risk-driven requirements implementation sequence algorithm used in agile planning.
- Visualize proposed agile plans based upon business value as well as two newly proposed priority schemes.

III. NORMATIC: An Agile Modeling Tool for NFRs

A. Background

NORMATIC was built as a semi-automatic tool on top of its underlying foundational framework, namely, the NORMAP Methodology. The NORMAP Methodology is composed of semi-automatic support for modeling NFRs in agile processes (NORMATIC) as well as a manual process (not addressed in this paper).

In an effort to identify, link, and model NFRs with FRs in agile processes, a number of enhancements had to be made. First, the W^8 User Story Card Model is proposed as an enhancement of the agile index card technique used to gather high-level agile requirements in XP and Scrum. The story card is a physical 3x5 index card used to capture 3-5 sentences of high-level requirements from the stakeholders' point of view. Despite the recommendation of [14] that requirements should be expressed in one sentence and avoid complex sentences (which may contain more than one requirement), capturing a user requirement in 1-2 sentences will almost certainly not suffice. On the other hand, the traditional use case model is heavyweight and stifles agile processes which value working software over documentation. A good balance between the two approaches was to capture the most important elements of a user requirement, along with essential project management and requirements quality metrics and automate as much of the NORMAP framework as possible in an easy to use visual environment. Furthermore, the tool supports risk calculations of NFRs and their potential solutions as well as visually reasoning about the requirements implementation sequence, hence, aiding in release and sprint planning.

A newly proposed Agile Requirement (AR) taxonomy was classified as either functional (Agile Use Case) or non-functional (Agile Loose Case). An Agile Choose Case is also part of the proposed taxonomy and represents the potential solutions (operationalizations) of NFRs. The three artifacts, AUCs, ALCs, and ACCs, form the three fundamental building blocks of the NORMAP Methodology around which the NORMATIC tool was developed.

In the W^8 Story Card model, clouds represent Agile Loose Cases, which model non-functional requirements (soft goals), or Agile Choose Cases, which model the soft goals operationalizations according to Chung's NFR Framework [13]. The light-colored non-bold cloud represents an ALC (green for source code such as an Aspect, red for architectural NFR, and blue for organizational policy NFR) while the dark-colored bold cloud represents an ACC. An Agile Choose Case—which is adapted from Chung's framework—models one or more potential solutions to the non-functional requirement. Our novel scheme suggests that potential solutions can be source code (e.g. an Aspect) which is color-coded as a dark green cloud, architectural or design constraint which is color-coded as a dark red cloud, or organizational policy which is color-coded as a dark blue cloud.

The following sections address fundamental modules in NORMATIC. Figure 1 shows NORMATIC's main screen.

B. Agile Use Case Details

The Agile Use Case (AUC) details screen includes requirement quality attributes (Sashimi-Complete or not, ambiguous or not, validated/completed or not). Other information includes the requested start/end dates, the release and sprint in which this requirement is requested, and an automatically-calculated risk score. NORMATIC automatically calculates how many times a requirement has changed

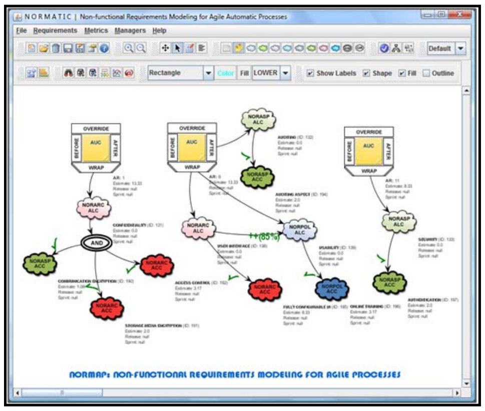

NORMATIC | Non-functional Requirements Modeling for Agile Automatic Processes
File Requirements Metrics Managers Help
Default
Rectangle
Color
Fill
LOWER
Show Labels
Shape
Fill
Outline
OVERRIDE
BEFORE
AUC
AFTER
WRAP
NORARC ALC
AND
NORASP ACC
NORARC ACC
NORASP ALC
NORPOL ALC
NORPOL ACC
++(85%)
CONFIDENTIALITY (ID: 131)
COMMUNICATION ENCRYPTION (ID: 190)
STORAGE MEDIA ENCRYPTION (ID: 191)
AUDITING (ID: 132)
AUDITING ASPECT (ID: 194)
USER INTERFACE (ID: 136)
ACCESS CONTROL (ID: 192)
USABILITY (ID: 139)
FULLY CONFIGURABLE UI (ID: 195)
ONLINE TRAINING (ID: 196)
SECURITY (ID: 133)
AUTHENTICATION (ID: 197)
NORMAP: NON-FUNCTIONAL REQUIREMENTS MODELING FOR AGILE PROCESSES

Figure 1. NORMATIC's Main Screen

(required for calculating requirement volatility risk). Estimates are entered in Fibonacci User Story Points (USPs) and the overall estimate is calculated according to the following well-known PERT Estimate formula (Estimate = [P + 4M + O] / 6, where P=Pessimistic, M=Most-Likely, O=Optimistic). Further, NORMATIC also captures the confidence factors for each of the provided estimates entered by the requirements engineer and automatically calculates a PERT confidence factor (also based on the above formula).

The W^8 User Story Card Model has eight "W"'s that capture the most essential and basic information of an agile user story, such as the "who", "what", "why", "when", etc. The W^8 User Story Card Model is captured either manually or through an automatic parsing of the full user story. The automatic parsing of the user story uses the Stanford Parser package and parses information only for the first 4 elements of the W^8 User Story Card Model. The first element captures the actor (required), the second one captures the action (required), the third element captures the reason for performing the action (optional), and the fourth one captures linguistically significant terms that may or may not indicate some non-functional requirements. The parsing stage does not perform the classification of this user story. The parsing is merely a mechanical process using the Stanford Parser to break a full user story into the first four elements of the W^8 model. In multi-sentence requirement statements, the requirements engineer is prompted to review each parsed sentence, and either accepts the parsed sentence as the main user story for this AUC or skips to the next sentence. The fifth element in the W^8 model captures optional quality attributes that are of lesser importance (e.g., optional NFRs). The sixth element captures the initial time frame within which the desired functionality is needed. The seventh element captures the requirement's initial priority from a business value standpoint. The eighth element captures a list of requirements dependencies impacted by this user story.

C. *Agile Loose Case Details*

An Agile Loose Case is a specific instance of an identified NFR for a particular user story. The term "Loose" was derived from the notion that NFRs are "soft" goals according to Chung's NFR Framework [13] and that such goals are usually hidden and difficult to identify (i.e. ill-defined or 'loose'). An ALC models not only a high-level NFR, but it also links to one or more Agile Choose Cases, which represent potential solutions (code-level aspects, design constraints and architecture, or organizational policy).

The Agile Loose Case (ALC) Details screen includes the same information already discussed in the AUC Details screen. The only additional items include an indicator whether this ALC is a mandatory or an optional one, and a priority field which indicates the business value priority requested by the Product Owner (which in most cases may be totally different from the technical priority based on requirements dependencies, architecture, and technical risks). The ALC risk field is calculated automatically based on the aggregate of all risks of the associated ACCs (potential solutions) plus the risks associated with the ALC itself (if it exists).

The number of changes, estimates, and estimates confidence are handled exactly the same way as an AUC. The ALC Details screen also shows the list of categories available (Non-functional Requirement Aspect or NORASP, Non-functional Requirement Architecture or NORARC, Non-functional Requirement Policy or NORPOL, and Untyped) as well as the list of all available ALCs that have already been entered into NORMATIC (25 NFRs that were selected for this study). The "Attributes" section displays and allows editing of all the relevant attributes that have been created and associated with this ALC using the Attributes Manager (discussed later).

D. *Agile Choose Case Details*

An Agile Choose Case represents one or more instances of potential solutions for the identified ALC. An ACC models not only the potential solution (i.e., operationalization according to Chung's NFR framework [13]), but it also captures other basic project management information such as relative size estimate (effort), estimate confidence, risk of implementing the solution (automatically calculated in real-time), business priority, whether the potential solution is a preferred one or not, and others.

The Agile Choose Case Details screen includes already stored potential solutions for non-functional requirements from which a requirements engineer can "choose" the most appropriate ones. The ACC Details screen includes the same information already discussed in the AUC Details screen. The only additional item is a preference indicator as to whether or not this solution is a preferred one, hence "chosen", over other possible solutions. If the solution is a "preferred" one, then this solution, and its associated risk, will be calculated as part of the aggregate risk for the ALC it is linked to. Otherwise, its risk will not impact its associated ALC, and will not be used in calculating the requirements implementation sequence. The ACC available categories are similar to the ALC ones available in the ALC Details screen. The "Attributes" section displays all the relevant attributes that have been created and associated with this ACC using the Attributes Manager.

E. *Requirements Quality Metrics Manager*

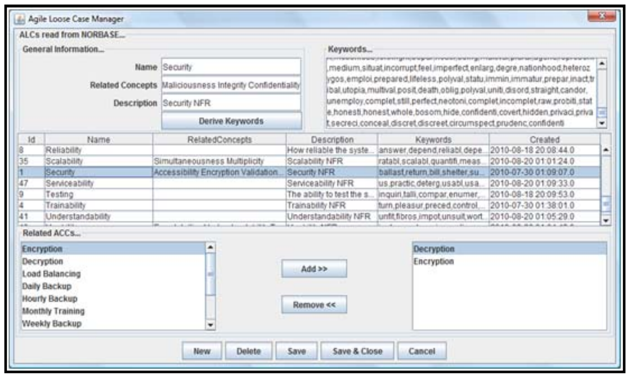
NORMATIC captures important requirements quality metrics that are used in calculating per-requirement risks as well as overall aggregate risks. Calculating risk is an integral part of the Requirements Implementation Sequence plan (NORPLAN) of the NORMAP Methodology. Metrics related to requirements quality include the total number of requirements (FRs/NFRs), ambiguity factor (0-1), completeness and validation factor (0-1), AUC and ALC densities, requirements volatility, and Agile maturity factors.

F. *Project Management Metrics Manager*

NORMATIC supports basic agile project management metrics which are used in computing a Requirements Implementation Sequence (NORPLAN). True and realistic requirements engineering cannot be achieved in isolation from project management. Effective agility blends both engineering as well as project management best practices. Therefore, achieving agility requires a systematic approach for calculating requirements quality as well as project management metrics. NORMATIC implements not only the metrics required for project management, but also the impact that such metrics would have on the requirements implementation sequence. An "Impact" metric captures how important a given metric is to the NORPLAN algorithm. For example, if "Team Development Factor" is not important (i.e., has no impact), then the impact value should be set to zero. If it has a moderate impact, say 10% impact, then its impact should be set to 10. The total value of all impacts must be equal to 100. NORPLAN was then used to generate a visual representation of the final implementation sequence plan in a tree-like view (NORVIEW).

G. *Agile Loose Case Manager*

The Agile Loose Case Manager (Figure 2) was used to manage (add/update/delete) metadata of ALCs in NORMATIC. Based on the name of an ALC, NORMATIC implemented a feature that derived all possible keywords related to this ALC using WordNet. NORMATIC used the name of the ALC to retrieve antonyms, synonyms, hypernyms, and hyponyms. In some instances, the name of the ALC was insufficient to derive quality keywords. Therefore, the "Related Concepts" field was introduced to allow a requirements engineer to optionally add space-delimited words that were related to an ALC. The "Related Concepts" field helped NORMATIC derive more keywords. NORMATIC then obtained the roots of those related words and stored them in the database (NORBASE). The ALC Manager also handled all related ACCs for a given ALC. The requirements engineer could add or remove ACCs related to a given ALC.



Name: Security
Related Concepts: Maliciousness Integrity Confidentiality
Description: Security NFR

,medium,situat,incorrupt,feel,imperfect,enlarg,degre,nationhood,heterozygos,emploi,prepared,lifeless,polyval,statu,immin,immatur,prepar,inact,tribal,utopia,multival,posit,death,oblig,polyval,uniti,disord,straight,candor,unemploy,complet,still,perfect,neotoni,complet,incomplet,raw,probiti,state,honesti,honest,whole,bosom,hide,confidenti,covert,hidden,privaci,privat,secreci,conceal,discret,discreet,circumspect,prudenc,confidenti

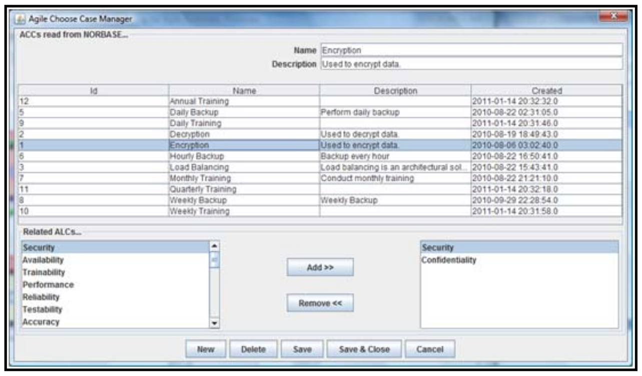
Id	Name	RelatedConcepts	Description	Keywords	Created
8	Reliability		How reliable the syste...	answer,depend,reliabl,depe...	2010-08-18 20:08:44.0
35	Scalability	Simultaneousness Multiplicity	Scalability NFR	ratabl,scalabl,quantifi,meas...	2010-08-20 01:01:24.0
1	Security	Accessibility Encryption Validation...	Security NFR	ballast,return,bill,shelter,su...	2010-07-30 01:09:07.0
47	Serviceability		Serviceability NFR	us,practic,deterg,usabl,usa...	2010-08-20 01:09:33.0
9	Testing		The ability to test the s...	inquiri,talli,compar,enumer,...	2010-08-18 20:09:53.0
4	Trainability		Trainability NFR	turn,pleasur,preced,control,...	2010-07-30 01:38:01.0
41	Understandability		Understandability NFR	unfit,fibros,impot,unsuit,wort...	2010-08-20 01:05:29.0

Encryption	Decryption
Decryption	Encryption
Load Balancing	
Daily Backup	
Hourly Backup	
Monthly Training	
Weekly Backup	

Figure 2. Agile Loose Case Manager

H. *Agile Choose Case Manager*

The Agile Choose Case Manager (Figure 3) was used to manage (add/update/delete) the metadata of ACCs in NORMATIC. This manager also handled all related ALCs for a given ACC. The requirements engineer could add or remove ALCs related to a given ACC. This was where an ACC (potential solution) was linked to an ALC (agile non-functional requirement). The actual details of how to implement concrete instances of an ACC was stored in a GraphML (XML) file as part of the data stored for every model. For example, a general ACC called "Encryption" was created in this screen. But the actual details for how encryption would be implemented for a specific requirement was stored in the XML file that described the specific requirements for that model. For example, one software system might be required to implement encryption using Triple-DES encryption algorithm while another system might be required to implement encryption using the Blowfish algorithm. Every agile model in NORMATIC was stored in its own GraphML (XML) file.



Name: Encryption
Description: Used to encrypt data.

Id	Name	Description	Created
12	Annual Training		2011-01-14 20:32:32.0
5	Daily Backup	Perform daily backup	2010-08-22 02:31:05.0
9	Daily Training		2011-01-14 20:31:46.0
2	Decryption	Used to decrypt data.	2010-08-19 18:49:43.0
1	Encryption	Used to encrypt data.	2010-08-06 03:02:40.0
6	Hourly Backup	Backup every hour	2010-08-22 16:50:41.0
3	Load Balancing	Load balancing is an architectural sol...	2010-08-22 15:43:41.0
7	Monthly Training	Conduct monthly training	2010-08-22 21:21:10.0
11	Quarterly Training		2011-01-14 20:32:18.0
8	Weekly Backup	Weekly Backup	2010-09-29 22:28:54.0
10	Weekly Training		2011-01-14 20:31:58.0

Security	Security
Availability	Confidentiality
Trainability	
Performance	
Reliability	
Testability	
Accuracy	

Figure 3. Agile Choose Case Manager

I. *Release and Sprint Managers*

The Release Manager handled basic information about releases, such as the name, start and end dates, and which sprints (iterations) were related to a given release. NORMATIC used this information to set a start date at which requirements implementation sequence (NORPLAN) was scheduled. However, NORMATIC will not honor the requested start/finish dates of a requirement if it was not inline with the calculated sequence based on the priority scheme selected.

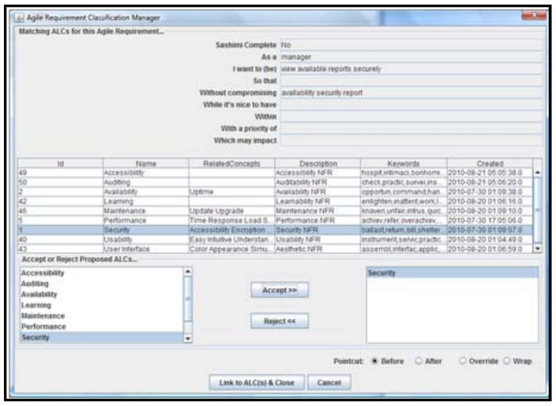
The Sprint Manager handled basic information about sprints, such as the name, start and end dates, and which releases are related to a given sprint. Again, if a given requirement is requested in a given sprint, it is up to the NORPLAN algorithm to decide whether risk calculations can honor the requested start date or not.

J. *Attributes Managers*

The Agile Attributes Manager handles all attributes that further describe either an ALC or an ACC. The manager stores the metadata information of each attribute and not the actual values. Actual values are stored in XML format. For example, the "Maximum Time Between Failure" attribute is used to further describe the "Availability" ALC as well as two ACCs, Daily Backup and Weekly Backup. An attribute may be used to describe either ALCs only, or ACCs only, or both.

K. Agile Requirements Classification Manager

When the user selects an agile requirement from the drawing pane and clicks on the Requirements Classification Manager, NORMATIC retrieves the information of the agile requirement, including the W^8 User Story Card Model and attempts to immediately classify the requirement (Figure 4). This classification involves identifying all possible non-functional requirements (ALCs). If found, ALCs will be retrieved, displayed, and the requirements engineer will have to either accept or reject the proposed ALCs. The user will also have to select a "Pointcut"—one of four possible locations (Before, After, Override, and Wrap) at which the new ALC will be linked to this requirement. Upon acceptance of the proposed ALC and selecting the appropriate pointcut, NORMATIC will link this requirement to the selected ALC and will visually create the correct ALC, place it on the drawing pane, and link the newly created ALC to the AUC at the correct "weaving" point.


Agile Requirement Classification Manager
Matching ALCs for this Agile Requirement...
Accept or Reject Proposed ALCs...
Accept >>
Reject <<
Link to ALC(s) & Close
Cancel

Sashimi Complete	No
As a	manager
I want to (be)	view available reports securely
So that	
Without compromising	availability security report
While it's nice to have	
Within	
With a priority of	
Which may impact	

Id	Name	RelatedConcepts	Description	Keywords	Created
49	Accessibility		Accessibility NFR	hospit,intimaci,bonhomi...	2010-08-21 05:05:38.0
50	Auditing		Auditability NFR	check,practic,survei,ins...	2010-08-21 05:06:20.0
2	Availability	Uptime	Availability NFR	opportun,command,han...	2010-07-30 01:09:38.0
42	Learning		Learnability NFR	enlighten,inattent,work,l...	2010-08-20 01:06:16.0
46	Maintenance	Update Upgrade	Maintenance NFR	knaven,unfair,intrus,quic...	2010-08-20 01:09:10.0
5	Performance	Time Response Load S...	Performance NFR	achiev,refer,overachiev,...	2010-07-30 17:05:06.0
1	Security	Accessibility Encryption ...	Security NFR	ballast,return,bill,shelter...	2010-07-30 01:09:07.0
40	Usability	Easy Intuitive Understan...	Usability NFR	instrument,servic,practic...	2010-08-20 01:04:49.0
43	User Interface	Color Appearance Simu...	Aesthetic NFR	assembl,interfac,applic,...	2010-08-20 01:06:59.0

Accessibility
Auditing
Availability
Learning
Maintenance
Performance
Security

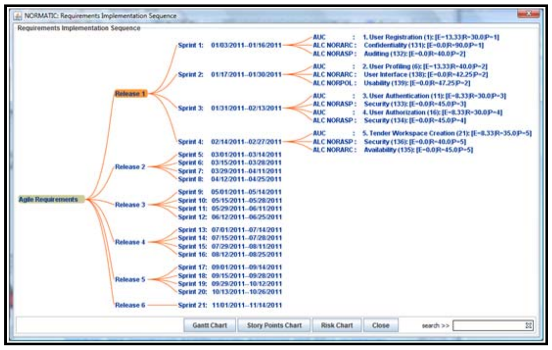
Security

Pointcut: [✓ Before] After Override Wrap

Figure 4. Agile Requirements Classification Manager

L. Agile Requirements Implementation Sequence Manager

The Requirements Implementation Sequence is the NORMAP Methodology's proposed plan of implementation (NORPLAN), which is based upon risk-driven calculations and the priority scheme selected for planning. As shown in Figure 5, the recommended implementation sequence plan is displayed in the form of a tree (NORVIEW) viewed from top to bottom with early requirements displayed at the top of the tree. The tree expands from left to right. The tree is composed of a root node, followed by the "Release" level, which in turn is followed by the "Sprint" (iteration) level. At the sprint level, each node displays the Start and End Dates next to the sprint label. Under every sprint node, the specific requirements and potential solutions (ACCs) will be displayed with Estimate (E), Risk (R), and Business Priority (P) parameters. For AUCs, the tree simply stops at this level since the NORMAP Methodology is not concerned with the details of implementing any particular functional requirements. For ALCs, the tree further expands to include the actual solutions to be implemented (ACCs). The Requirements Implementation Sequence Manager also has options to view a traditional Project Management Gantt Chart and two bar charts that show user story points implemented as well as calculated risks in every sprint.



```
Sprint 1:   01/03/2011..01/16/2011   AUC        :  1. User Registration (1): [E=13.33|R=30.0|P=1]
                                     ALC NORARC :  Confidentiality (131): [E=0.0|R=90.0|P=1]
                                     ALC NORASP :  Auditing (132): [E=0.0|R=40.0|P=2]
Sprint 2:   01/17/2011..01/30/2011   AUC        :  2. User Profiling (6): [E=13.33|R=40.0|P=2]
                                     ALC NORARC :  User Interface (138): [E=0.0|R=42.25|P=2]
                                     ALC NORPOL :  Usability (139): [E=0.0|R=47.25|P=2]
Sprint 3:   01/31/2011..02/13/2011   AUC        :  3. User Authentication (11): [E=8.33|R=30.0|P=3]
                                     ALC NORASP :  Security (133): [E=0.0|R=45.0|P=3]
                                     AUC        :  4. User Authorization (16): [E=8.33|R=30.0|P=4]
                                     ALC NORASP :  Security (134): [E=0.0|R=45.0|P=4]
Sprint 4:   02/14/2011..02/27/2011   AUC        :  5. Tender Workspace Creation (21): [E=8.33|R=35.0|P=5]
                                     ALC NORASP :  Security (136): [E=0.0|R=40.0|P=5]
                                     ALC NORARC :  Availability (135): [E=0.0|R=45.0|P=5]
Sprint 5:   03/01/2011..03/14/2011
Sprint 6:   03/15/2011..03/28/2011
Sprint 7:   03/29/2011..04/11/2011
Sprint 8:   04/12/2011..04/25/2011
Sprint 9:   05/01/2011..05/14/2011
Sprint 10:  05/15/2011..05/28/2011
Sprint 11:  05/29/2011..06/11/2011
Sprint 12:  06/12/2011..06/25/2011
Sprint 13:  07/01/2011..07/14/2011
Sprint 14:  07/15/2011..07/28/2011
Sprint 15:  07/29/2011..08/11/2011
Sprint 16:  08/12/2011..08/25/2011
Sprint 17:  09/01/2011..09/14/2011
Sprint 18:  09/15/2011..09/28/2011
Sprint 19:  09/29/2011..10/12/2011
Sprint 20:  10/13/2011..10/26/2011
Sprint 21:  11/01/2011..11/14/2011
```

Figure 5. Agile Requirements Implementation Sequence Manager

IV. Case Studies: the European Union Electronic Procurement System and the PROMISE Dataset

A. Background

NORMATIC was validated using two case studies. The first case study utilized the requirements model of the European Union (EU) eProcurement Online System [17]. The requirements in the EU case study were inter-related and represented one coherent system. However, the validation process in this case study focused primarily—among other aspects such as parsing and NFR classification—on visually identifying and linking AUCs to ALCs and their corresponding potential solutions (ACCs). NORMATIC was also used to compute and visualize an improved risk-driven agile requirements implementation sequence given different priority schemes, risk factors, and the metrics and impacts associated with requirements. The second case study utilized the Predictor Models in Software Engineering (PROMISE) repository of empirical software engineering dataset and classified 607 independent user requirements using a special batch process [18].

B. eProcurement System Requirements

The EU case study [17] included 26 functional requirements from which other non-functional requirements were identified. NORMATIC was fully or partially successful in identifying 16 non-functional requirements given only the detailed description of all 26 functional requirements. The following 16 ALCs were identified: Confidentiality, Security, Auditability, Usability, User Interface, Accessibility, Availability, Multilingual Support, Documentation, Accuracy, Compliance, Efficiency, Performance, Interoperability, Scalability, and Configurability.

NORMATIC parsed each sentence stored in every AUC and prompted the requirements engineer to either accept or reject the parsed sentence as the main user story, hence, ignoring unnecessary additional information. ALCs were identified and linked to AUCs and potential solutions (ACCs) were selected and linked to their corresponding ALCs.

C. Building the Initial Model for the eProcurement System

Modeling the eProcurement system's functional and non-functional requirements and their potential solutions using the new agile artifacts and NORMATIC is summarized in Table I. Some steps include details that are outside the scope of this paper and will be addressed in separate future papers.

TABLE I. MODELING AGILE REQUIREMENTS USING NORMATIC

Step	***Description***
1	Selecting and entering the initial 25 selected NFRs, deriving keywords for each NFR, and storing such meta-data information into NORMATIC's meta-database (NORBASE).
2	Associating ALCs (NFRs) to their ACCs (potential solutions) in the ALC and ACC Managers, respectively.
3	Configuring project management and setting up necessary requirements quality metrics.
4	Estimating relative efforts (Fibonacci-based) and confidence factors (used in part to calculate risks).
5	Manually entering all 26 functional requirements as 26 Agile Use Cases. For each AUC, the entire original description of each requirement statement was entered, parsed, and processed by the system. In multi-sentence stories, only one sentence was selected as the main user story.
6	Identifying and classifying possible Agile Loose Cases for each of the 26 Agile Use Cases.
7	Linking identified ALCs with their corresponding AUCs as well as linking the ALCs with their corresponding ACCs (potential solutions). Each AUC has four "weaving points" (pointcuts): *Before*, *After*, *Override*, and *Wrap*. Depending on the context and understanding of each requirement, ALCs were connected to the appropriate pointcuts.
8	Additional steps (outside the scope of this paper) were carried out, such as, conducting three experiments using different proposed priority schemes, calculating and updating risk scores for every requirement, and visualizing an improved requirements implementation sequence. Details of those additional steps and results will be reported in future papers.

D. First Case Study: Agile Requirements Modeling Example

Figure 6 represents the modeling of Requirement 12 of 26 in the eProcurement system. The functional requirement is represented as an AUC (yellow sticky note icon surrounded by four pointcuts). Each AUC is linked to one or more ALC and each ALC is linked to one or more ACC.

The AUC pertains to the publication of contract documents—a functional requirement that provides the ability to model internal workflows (e.g., creation, validation, approval, and publication). This functionality can assist a Procurement Officer to comply with internal workflows of their contracting authority in a more efficient and time-effective manner. Furthermore, the requirement stipulates that while a document is in "not-published" status, it should be made available only to authorized Procurement Officers associated with it and not made publicly available. The document is made publicly available only after the Contract Notice is sent for publication. In this requirement, NORMATIC identified a number of Agile Loose Cases (NFRs), such as "Security" (light-green cloud), "Availability" (light-red cloud), "Documentation" (light-green cloud), "Compliance" (light-blue cloud), and "User Interface" (light-red cloud). The Security and User Interface (ease of use) ALCs were connected to the "Wrap" pointcut as both NFRs must be satisfied before and after the core functional behavior of publishing contract documents is carried out.

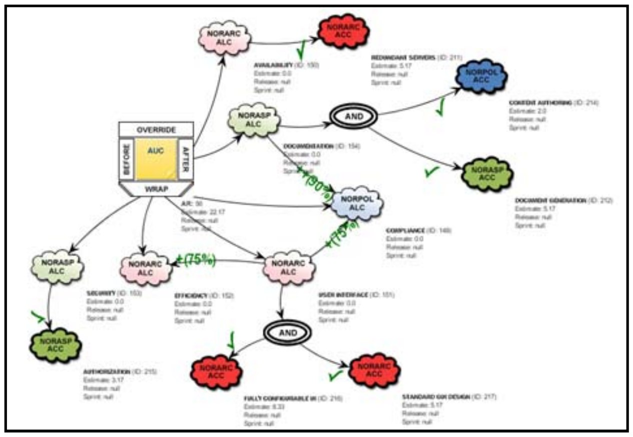


Figure 6. Modeling of Publication of Contract Documents

Security can include potential solutions such as Authorization (dark-green cloud). Authorization may require source-code level validation of a user's role, which calls for modeling it as NORASP ACC. NORASP ACC is typically implemented as an Aspect (from the field of Aspect-Oriented Software Development). In case of the ease of use for the User Interface, potential solutions may include standard GUI design practices and fully-configurable user interface (both are modeled as dark-red clouds). Configurable GUI interface may potentially be thought of as an architectural and design constraint, therefore, potential solutions were modeled as NORARC ACCs (dark-red clouds). It should be noted that because both potential solutions are required to satisfy the User Interface NFR, the "AND" operator was used to connect both solutions to the User Interface ALC. Another example of a potential architectural solution is the "Redundant Servers" ACC (dark-red cloud) which is directly linked to the "Availability" ALC (light-red cloud).

In Figure 6, the "Documentation" ALC can have a very positive impact on achieving the "Compliance" ALC as well. This was modeled by the edge that is labeled with "++(90%)". The "Documentation" ALC itself can be potentially solved through an organizational policy for "Content Authoring" (which is modeled as a dark-blue cloud indicating an internal organizational process for authoring content) and automatic "Document Generation" (which is modeled as a dark-green cloud indicating the possibility of "weaving" an Aspect at the "After" pointcut to automatically generate the documents).

Once all requirements modeling was completed, risk calculations were carried out and NORMATIC computed a risk-driven requirements implementation sequence (i.e., NORPLAN or simply the agile project plan) based upon requirements quality and project management metrics. The plan is then visualized in a tree-like view (called NORVIEW) as shown in Figure 5.

E. Second Case Study: The PROMISE Dataset

The second case study utilized the Predictor Models in Software Engineering (PROMISE) repository of empirical software engineering dataset to classify NFRs. The focus of the second case study was to evaluate NORMATIC's ability to parse requirements statements and identify possible Agile Loose Cases (NFRs). This case study utilized a special NORMATIC batch process to parse and classify 607 independent system requirement statements. No visualization features were used in this case study. The 607 independent requirement statements did not represent one coherent system. Rather, the dataset represented a collection of independent requirement statements with their pre-determined requirement types from 15 different projects in a graduate-level software engineering course at DePaul University. Therefore, it did not make sense to utilize NORMATIC's visual capabilities to model unrelated individual requirements.

F. Results and Analysis

In first case study, NORMATIC was used to model the requirements of one coherent system—the eProcurement System—with its 26 functional requirement statements with well over 100 sentences of requirement descriptions. Each functional requirement statement was modeled as an Agile Use Case (modeled as yellow sticky note with four pointcuts).

NORMATIC was successful in fully or partially parsing and classifying 50 out of 57 user requirement sentences that contained non-functional requirements—an overall success rate of 87.71%. Further, NORMATIC identified the following 16 ALCs: Confidentiality, Security, Auditability, Usability, User Interface, Accessibility, Availability, Multilingual Support, Documentation, Accuracy, Compliance, Efficiency, Performance, Interoperability, Scalability, and Configurability. Each ALC was modeled as a light-colored non-bold cloud with three possible colors: light-green, light-red, and light-blue. Light-green modeled ALCs that can potentially be solved by a source code solution (e.g., Aspects in Aspect-Oriented Software Development). Light-red clouds modeled ALCs that can potentially be solved by architectural and design constraints (e.g., load balancing hardware for Performance ALCs). Light-blue clouds modeled ALCs that can potentially be solved by organizational policies (i.e., non-technical).

Potential solutions (operationalizations) were modeled as Agile Choose Cases and were modeled as dark-colored bold clouds with three possible colors: dark-green (such as Aspects), dark-red (architectural decisions), and dark-blue (organizational policy or non-technical solutions).

NORMATIC currently supports linking artifacts in three different configurations: AUC-to-ALC, ALC-to-ACC, and ALC-to-ALC. In the first case, a functional requirement is linked to a non-functional requirement. The second case links a non-functional requirement to a potential solution. In the third case, a non-functional requirement is positively or negatively impacting another non-functional requirement (impact value ranges from 0 which is severe negative impact to 100 which is absolutely positive impact). The value of impact between any two ALCs is one of many factors used to calculate a composite risk score for any given ALC.

NORMATIC's fundamental design included the ability to visually model functional and non-functional requirements and their potential solutions in agile processes such as Scrum. The different color coding scheme could potentially improve agility in visually distinguishing between source code-based NFRs and solutions (green clouds), architectural and design constraints NFRs and their solutions (red clouds), and non-technical organizational policies (blue clouds). Furthermore, adapting the concept of pointcuts (Before, After, Override, Wrap) from the field of Aspect Orientation and utilizing it to visually model the instant of time at which an Agile Loose Case is linked (or weaved) to an Agile Use Case was visually effective. Agility was also achieved through utilization of Natural Language Processing (NLP) tools that were used to parse and classify ALCs, and visualization tools that were used to model risk-based requirements implementation sequence (i.e., the agile project plan). In addition, project management and requirements quality metrics were all used to achieve agility without compromising a lightweight engineering process.

In summary, NORMATIC has many features that can potentially help agile teams reason about NFRs in an agile environment. Rapid identification of NFRs and linking them to their functional counterparts allow NFRs to be treated as first-class artifacts early on while providing light-weight yet agile engineering process that is systematic and repeatable. Project managers can potentially benefit from NORMATIC through its risk-driven requirements implementation sequence computation and visualization capabilities. Future research papers will shed more light on other aspects of this tool and its potential use.

V. Conclusion and Future Work

This research addressed NORMATIC as a viable modeling tool that simulated the underlying NORMAP Methodology. NORMATIC can easily model functional requirements, non-functional requirements, and non-functional requirements operationalizations (potential solutions) in a visual environment that can be used in agile processes, such as Scrum. This research study is part of the NORMAP Methodology which is concerned with the identification, modeling, and linking of agile NFRs (Agile Loose Cases) and their potential solutions (Agile Choose Cases) with their functional counterparts (Agile Use Cases). An Agile Use Case is modeled as a yellow sticky note surrounded by four "weaving" points (i.e., pointcuts in Aspect-Oriented Software Development), namely, *Before*, *After*, *Override*, and *Wrap*. An Agile Loose Case is modeled as a light-colored cloud with three possible colors: light-green for source code type ALC (e.g., an Aspect), light-red for architectural and design constraint type ALC, and light-blue for organizational policy type ALC. An ALC can be linked to an AUC through one of the four pointcuts of an AUC. An Agile Choose Case is modeled as a dark-colored bold cloud with three possible colors: dark-green for source code type ACC, dark-red for architectural and design constraints type ACC, and dark-blue for organizational policy type ACC.

NORMATIC, a visual modeling simulation tool, was designed and developed in a Java-based environment that utilized visualization frameworks, such as JUNG [19] and Prefuse [20] as well as Natural Language Processing tools for parsing and classifying requirements.

The EU eProcurement System Requirements Model was particularly effective in visualizing and modeling functional and non-functional requirements dependencies, how ALCs can positively or negatively impact other ALCs as well as AUCs, and how ALCs can be linked to their corresponding ACCs. In addition, the development of the visual framework tremendously helped in visually differentiating between functional and non-functional requirements as well as potential solutions with different shapes and color-coded cloud icons which can all be used in agile processes, such as Scrum. The second case study utilized a special batch process to parse and classify NFRs from independent requirement statements and did not benefit from NORMATIC's visualization capabilities.

This research study hypothesized that agile processes can still benefit from lightweight engineering processes that can capture NFRs as first-class artifacts early on, and not treating them as an afterthought process. Visualization tools were used to develop the modeling and simulation tool that models primarily agile non-functional requirements (ALCs) and their potential solutions (ACCs), as well as agile functional requirements (AUCs) in an easy to understand and visually appealing environment without compromising agility. Scrum teams and project managers can potentially benefit from NORMATIC's requirements quality and project management metrics, which were used to calculate a risk-driven requirements implementation sequence (the plan) and visualize it as a tree.

The results of this research highlight the need to validate NORMATIC in real-world agile development projects and measure the effectiveness of using the tool in modeling NFRs in an agile and Scrum-like setting. Other improvement areas include providing a web-enabled version of NORMATIC, supporting mobile computing environments, and service-oriented architecture-based agile modeling services.

REFERENCES

[1] G. Goth, "Agile Tool Market Growing with the Philosophy", 2009, *IEEE Software,* 26(2), pp. 88-91.

[2] A. Sampaio, N. Loughran, A. Rashid, and P. Rayson, "Mining Aspects in Requirements", *In Early Aspects Workshop, International Conference on AOSD*, 2005.

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

[4] 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.

[5] M. Qasaimeh, H. Mehrfard, and A. Hamou-Lhadj, "Comparing Agile Software Processes Based on the Software Development Project Requirements", *In Proceedings of 2008 International Conferences on Computational Intelligence for Modeling, Control and Automation; Intelligent Agents, Web Technologies and Internet Commerce; and Innovation in Software Engineering*, 2008, pp. 49-54.

[6] 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.

[7] 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.

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

[9] M. Woodside, G. Franks, and D. Petriu, "The Future of Software Performance Engineering", *In Future of Software Engineering (FOSE '07)*, 2007, pp. 171-187.

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

[11] 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.

[12] A.M. Christie, "Simulation in support of CMM-based process improvement", *Journal of Systems and Software* (46), 1999, pp.107-112.

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

[14] S. Robertson and J. Robertson, *Mastering the Requirements Process*, ACM Press Books, Addison-Wesley, Harlow et al., 1999.

[15] J. Araújo and M. Moreira, "An Aspectual Use-Case Driven Approach", *In VIII Jornadas Ingeniería del Software y Bases deDatos (JISBD 2003)*, 2003, pp. 463-468.

[16] 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.

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

[18] G. Boetticher, T. Menzies, and T. Ostrand, PROMISE Repository of Empirical Software Engineering Data, West Virginia University, Department of Computer Science, 2007, http://promisedata.org/?p=38.

[19] JUNG Development Team, "The Java Universal Network/Graph Framework", Internet: http://jung.sourceforge.net, 2003 [May 15, 2010].

[20] Prefuse Development Team, "Prefuse: The Information Visualization Toolkit", Internet: http://prefuse.org, 2007 [May 17, 2010].