Journal of Software Engineering Practice, 2018 2(1)

Agile Software Engineering & the Future of Non-Functional Requirements

Richard R. Maiti
Department of Computer Science
Berea College
Berea, KY
maitir@berea.edu

Aleksandr Krasnov
Department of Computer Science
Berea College
Berea, KY
krasnova@berea.edu

Deanna Marie Wilborne
College of Engineering & Computing
Nova Southeastern University
Fort Lauderdale, FL
mw1313@mynsu.nova.edu

***Abstract*— Non-Functional requirements (NFRs) are overlooked whereas Functional Requirements (FRs) take the center stage in developing agile software. Research has shown that ignoring NFRs can have negative impact on the software and could potentially cost more to fix at later stages. This research extends the CEP methodology to predict NFRs in the next iterations of agile software development. Research in other fields have shown that historical data can be beneficial in the long run. This research shows that using the data available can be beneficial for the next iteration of software development. A simple decision tree was utilized to predict future NFRs based on past data. There are multiple occurrences of NFRs and security was found to have the most occurrences.**

Keywords—Agile Software Engineering, Capture Elicit Prioritize, CEP ,Functional Requirements, Non-Functional Requirements, NFRs, FRs, NERV, NORMAP, historical trending, historical NFRs, decision tree

INTRODUCTION

Functional Requirements (FRs), during development, are given higher ranking due to the characteristics of agile software development methodology. Non-Functional Requirements (NFR), which are behaviors of a system, are often ignored and are added towards the end of the software development cycle (Nguyen, 2009). Some behaviors of NFRs are the following: availability, capacity, data integration maintenance, performance, reliability, regulatory, scalability, security, and usability (Farid & Mitropoulos, 2012). The research of NFRs is gaining priority and more consideration is being placed on NFRs by giving NFRs the same importance as FRs. FRs are defined as functions of a system; NFRs, however, focus on the characteristics of the system (Ameller, 2012). The success of a software development system includes NFRs and FRs (Slankas & Williams, 2012). A successful software system is dependent on an agreement between the stakeholders and includes both FRs and NFRs (Danylenko & Lowe, 2012; Poort et. al, 2012). Mobile applications and software systems are increasingly complex; therefore, it is important to include NFRs during development (Yin & Jin, 2012). Not considering NFRs for software systems may result in a failure rate of 60% or higher (Fabio et. al, 2013; Bajapi & Gorthi, 2012).

Scrum, a framework for agile software development, relies on developing software quickly by considering FRs (Farid & Mitropoulos, 2012). NFRs are starting to be adopted in earlier processes of software engineering (Saadatmand et. al., 2012; Bajapi & Gorthi, 2012; Farid & Mitropoulos, 2012; Liu, 2012). Considering NFRs earlier in development can significantly reduce the number of defects (Saito et. al, 2012). By including NFRs and FRs concurrently during development, stakeholders can realize cost savings and better software.

Incorporating historical trending to predict NFRs can be beneficial to predict a future outcome based on historical data. Research has shown historical data is useful to determine a future event based on past data. The medical field uses summaries of past data when considering diagnosis for a patient (Salatian, 2009).

Research Goals and Research Questions

Research Goals

This research extends the Capture Elicit Prioritize (CEP) methodology to include a prediction model (Maiti & Mitropoulos, 2017). NFRs are often ignored due to the characteristics of agile software engineering. Some characteristics are daily meetings with team members to discuss what was completed the previous day, setting goals and discussing road blocks. These are known as Scrum meetings. Considering NFRs and including NFRs during the earlier process of agile software engineering, has benefits. This research extends the CEP methodology (Maiti & Mitropoulos, 2017) to use the historical data from the European Union (EU) procurement document to predict NFRs for the next iteration of software development (European Dynamics S.A., 2005a) and (European Dynamics S.A., 2005b).

Research Question

This research answers the following question:

RQ: Can historical metadata have an impact in predicting NFRs?

Brief Literature on Historical Trending

Medical staff members deal with big amounts of data that are noisy (Salatian, 2009). Short summaries of data can help patients in deciding what treatment needs to be taken. The research proposed by (Salatian, 2009), developed an algorithm for intervals in historical data where the attributes are *possible value increasing*, *decreasing* or *steady holds*, which are trends of data over the interval. The Wavelet algorithm process was used to look at data at different scales and resolutions (Salatian, 2009). The strength of the research shows that having an ample amount of data and being able to look at snap shots can be advantageous in predicting the next step. However, the weakness of this research lies on the amount of historical data that is available at a given time, which could be critical in this type of environment.

The research conducted by (Koomey et. al, 2011), shows that computer performance is growing steadily over the past 65 years. The performance of personal computers has doubled in performance every 1.5 years, which corresponds to Moore’s Law (Koomey et. al, 2011). The electrical efficiency also doubled every 1.5 years (Koomey et. al, 2011). The main trend found is increased efficiency and reduced cost, due to smaller transistor size, which explains the reduced usage of electricity and improved computational performance (Koomey et. al, 2011). The trends included laptop computers, cellphone and personal digital assistants. If the trends continue, this will reduce the power consumption of mobile devices and increase development of new applications for mobile computing, sensors and controls (Koomey et. al, 2011).

Methodology

The Capture Elicit Prioritize (CEP) methodology extended the NERV and NORMAP methodologies from previous research (Maiti & Mitropolous, 2015; Maiti & Mitropouls, 2017; Maiti & Mitropouls, 2017; Farid, 2011; Doomah, 2013). The CEP methodology identified 56 out of 57 requirement sentences and was successful in eliciting 98.24% of the baseline. This is an improvement of 10.53% over the NORMAP methodology, and 1.75% improvement over the NERV methodology (Maiti & Mitropoulos, 2017; Maiti & Mitropouls, 2017). The NFRs count for the CEP methodology was 86 out of 88 NFRs, which was an improvement of 12.49% over the NORMAP methodology and 4.55% over the NERV methodology(Maiti & Mitropoulos, 2017; Maiti & Mitropouls, 2017). The CEP used and utilized the EU eProcument requirements document (European Dynamics S.A., 2005a) and (European Dynamics S.A., 2005b).

The Capture component, of the CEP methodology used OCR to gather potential NFRs using requirements images (Maiti & Mitropoulos, 2015; Maiti & Mitropouls, 2017). (Maiti & Mitropoulos, 2017Then places the sentences into distinct categories by utilizing the *k*-NN classification algorithm (Slankas and Williams, 201). The NFR categories are defined from Chung's NFR framework utilizing a set of keywords for training, to locate NFRs (Maiti & Mitropoulos, 2017; Maiti & Mitropouls, 2017). The e αβγ-framework was utilized to prioritize the NFRs. This is a flexible framework that enables agile members to substitute other sub processes to prioritize the NFRs, and is the final component of the CEP methodology (Maiti & Mitropoulos, 2017; Maiti & Mitropouls, 2017).

Utilizing the data from previous research of the CEP methodology this research further extends the CEP methodology (Slankas and Williams, 2013). The extended research includes a decision tree to predict future NFRs. A simple decision tree was utilized to make a prediction using the past NFRs data. Figure 1 below helps visualize the prediction.

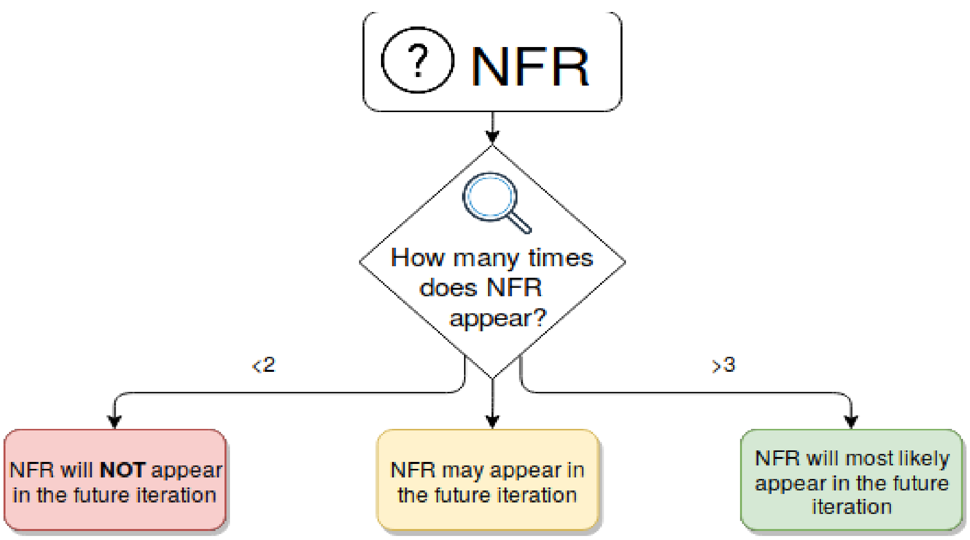

Figure 1. Decision Tree to make a prediction

As shown in Figure 1, if NFRs appears multiple times in the requirements document then it is most likely that NFRs will appear again in future requirements. If the NFRs is equivalent to three times, it is likely it will appear in the next iteration. If the NFRs is between one and two it is not likely to appear in a future iteration. The path can be traced from the root of the tree to a decision tree's leaf (yes or no) that determines whether the NFRs will appear in future iterations.

In Figure 2, is the update of the CEP model. The process of capturing, eliciting, and prioritizing is the original CEP model. The update incorporates the decision tree which is included in the new updated CEP model to make a prediction thus creating the Capture Elicit Prioritize and Predict (CEPP) model as shown in figure 2.

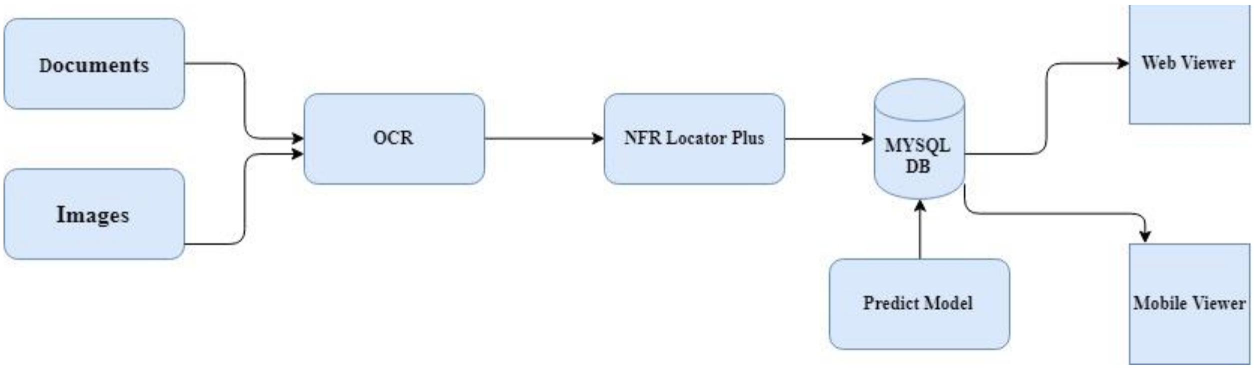

Figure 2. Updated CEP model incorporating Prediction

Results

This section covers the results that the CEP methodology. The European Procurement (EU) document volume 1 and volume 2 are system requirement documents that were used in this research as it has a solid background in previous research in NORMAP, NERV and CEP (Maiti & Mitropoulos, 2017; Maiti & Mitropouls, 2017; Farid, 2011; Doomah, 2013). These results were captured from the previous CEP research where each baseline NFRs was recorded and the number of times the NFR appeared in the EU procurement document (European Dynamics S.A., 2005a) and (European Dynamics S.A., 2005b) taking the baseline set of NFRs. The number of occurrences of NFRs are shown below in Figure 3.

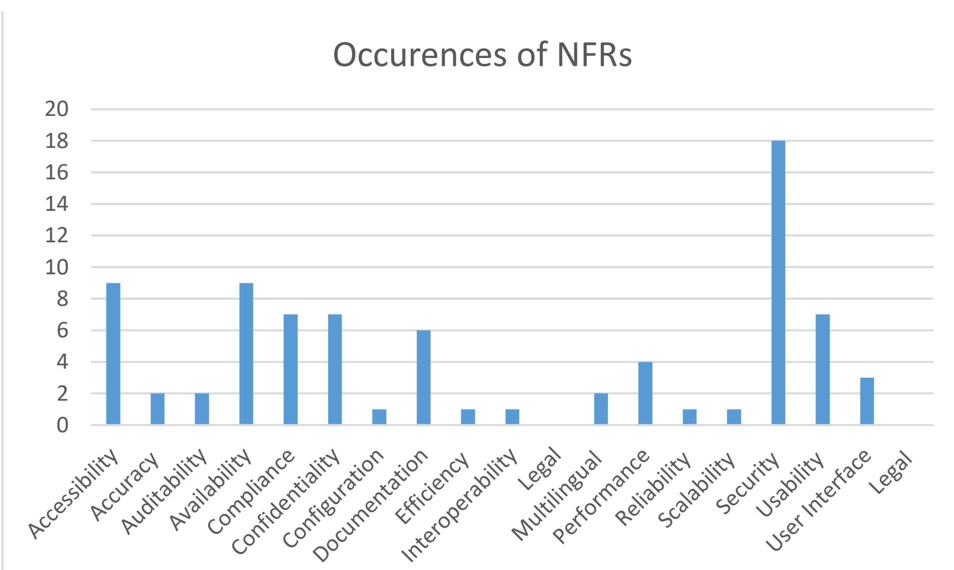

Figure 3. The number of occurrences of NFRs

Applying the decision tree, the NFRs that were predicted to be in future iterations are the ones that appeared 3 times or more. The NFRs that appear more than three times or more are the following: Accessibility, Availability, Compliance, Confidentiality, Documentation, Performance, Security, Usability and User Interface. These NFRs are likely to appear in the next iteration of the software development cycle. Furthermore, similar applications can have these NFRs present in their requirements documentation. The prediction of NFRs utilizing the decision tree is useful when creating the requirements documentation. Using historical data is useful in predicting future NFRs. Literature has shown that historical data can be useful in other fields and therefore it is also useful in the early process of agile software development.

Conclusion & Future Studies

This research investigated whether historical NFRs data were useful in the future iteration of software development. The following research question was raised:

RQ: Can historical metadata have an impact in predicting NFRs?

Previous research in other fields has shown that historical trending based on gathered historical metadata is beneficial. This research shows that historical metadata can help in predicting NFRs by utilizing a decision tree to make a prediction. NFRs that appear multiple times in a set of the EU procurement documents can be useful in predicting future NFRs. The NFRs Availability, Compliance, Confidentiality, Documentation, Performance, Security, and Usability were found multiple times using the previous CEP methodology research data (Maiti & Mitropoulos, 2017).

In agile software development process security is taken into consideration and fixed in an ad-hoc manner (Maiti & Mitropoulos, 2015; Maiti & Mitropoulos, 2017; Maiti & Mitropouls, 2017). Agile team can use the NFRs data to develop secure code by identifying vulnerabilities in the code ahead of time instead of fixing security in an ad-hoc manner.

For future studies, this research can be extended to identify historical data that appear in multiple occurrences that can be proved to be crucial NFRs. NFRs such as Security, which appears 18 times, can be crucial in developing code that is secure. These NFRs can be grouped as crucial NFRs and additional measures can be taken to develop secure software. This research took a small set of historical data to show that historical trending can be beneficial in predicting the next iteration of software engineering. More historical data is required to determine if historical NFRs can be beneficial in the long run.

References

Affleck, A. & Krishna, A. (2012). Supporting quantitative reasoning on non-functional requirement a process-oriented approach. Proceedings from ICSSP: *The International Conference on Software and System Process*, Zurich, Switzerland, 88-92.

Ameller, D., Ayala, C., Cabot, J., & Franch, X. (2012). How do software architect consider non-functional requirements: An exploratory study. Proceedings from RE: *The 20th International Requirements Engineering Conference*, Chicago, IL, 41-50.

Bajapi, V. & Gorthi, R. P. (2012). On non-functional requirements: A survey. Proceedings from ICCNIT: The International Conference on Computer Networks and Information Technology, Abbottabad, Pakistan, 333-340.

Danylenko, A. & Lowe, W. (2012). Context-aware recommender systems for non - functional requirement. Proceedings from RSSE: *The 3rd International Workshop on Recommendation Systems for Software Engineering*, Zurich, Switzerland, 80-84.

Domah, D. (2013). The NERV methodology: Non-functional requirements elicitation, reasoning and validation in agile processes (Doctoral Dissertation) Available from ProQuest Dissertation and Thesis database (UMI No. 3594275)

European Dynamics S.A. (2005). Functional Requirements for Conducting Electronic Public Procurement Under the EU Framework (Volume 1). on May 30, 2018 from http://ec.europa.eu/internal_market/publicprocurement/docs/eprocurement/functional-reguirements-vol1_en.pdf

European Dynamics S.A. (2005). Functional Requirements for Conducting Electronic Public Procurement Under the EU Framework (Volume 2). Retrieved on May 30, 2018 from http://ec.europa.eu/internal_market/publicprocurement/docs/eprocurement/functional-reguirements-vol2_en.pdf

Fabio, S., Lucena, M., and Lucena, L., (2013). STREAM-AP: A Process to systematize architectural patterns choice based on NFR. *The 3rd International Workshop on Twin Peaks of Requirement and Architecture*, Rio de Janeiro, Brazil, pp. 27-34.

Faird, W. M. (2011). The NORMAP methodology: Non-functional requirements modeling for agile processes (Doctoral Dissertation) Available from ProQuest Dissertation and Thesis database. (UMI No. 3460458)

Farid, W. M. & Mitropoulos, F. J. (2012). Novel lightweight engineering artifacts for modeling non-functional requirements in agile processes. Proceedings of IEEE: *Southeastcon*, Orlando, FL, 1-7.

Koomey, J.G., Berard, S., Sanchez, J., & Wong, H. (2011). *Implication of historical Trends in the electrical efficiency of computing*. Annals of the history of computing, 33(3), 46-54.

Liu, Y., Zhiyi, M., Qiu, R. Chen, H., & Shao, W. (2012). An approach to integrating non-functional requirements into UML design models based on NFR-specific patterns. Proceedings from QSIC: *The 12th International Conference on Quality Software*, Shaaxi, China, 132-135.

Maiti, R. R. & Mitropoulos, F. J. (2015). Capturing, eliciting, predicting and prioritizing (CEPP) non-functional requirements metadata during the early stages of agile software development. *In IEEE, (Southeastcon '15*), Ft. Lauderdale, FL, 2015, 1-8.

Maiti, R. R. & Mitropoulos, F. J. (2017). Capturing, eliciting, and prioritizing (CEP) NFRs in agile software engineering, *SoutheastCon*, Charlotte, NC, 2017, 1-7.

Maiti, R. R., & Mitropoulos, F. J. (2017) Prioritizing Non-Functional Requirements in Agile Software Engineering. In *Proceedings of the SouthEast Conference* (ACM SE '17). ACM, New York, NY, 212-214.

Nguyen, Q.L. (2009). Non-Functional Requirements analysis modeling for software product line. *Proceedings of the 2009 ICSE Workshop on Modeling in Software Engineering*, Washington, D.C., 56-61.

Poort, E. R., Key, A., With, P.H.N. & Vilet, H. (2012). Issues dealing with non-functional requirements across the contractual divide. Proceedings of *WICSA and ECSA: Software Architecture and European Conference on Software Architecture*, Helsinki, Finland, 315-319.

Saadatmand, M., Cicchetti, A., & Sjodin, M. (2012). Toward model-based trade-off analysis of non-functional requirements. Proceedings from SEAA : *The 38th Conference on EUROMICROS*, Cesme, Turkey, 142-149.

Saito, Y., Matsumoto, K., & Moden, A. (2012). Evaluation of non functional requirements in a Request For Proposal (RFP). Proceedings from IWSM- MENSURA: *The 22nd International Workshop on Software Measurements and the 7th International Conference on Software Process and Product Measurement*, Assisi, Italy, 106-111.

Salatian, A., Adepoju, F., & Odinma, A. (2009). *A data wavelets approach to deriving trends in historical ICU monitor data. Proceedings of Sensor Applications and Symposium*, Limerick, Ireland, 162-165

Slankas, J. & Williams, L. (2013). Automated extraction of non-functional requirement in available documents. Proceedings for NaturaLise: *The 1st International Workshop on Natural Language Analysis in Software Engineering*, San Francisco, CA, 9-16.

Yin, B. & Jin, Z. (2012). Extending the problem frames approach for capturing non –function requirements. Proceeding in *9th International Conference on Computer and Information Sciences*, Shanghai, China, 432-437.