B.Comp. Dissertation

Prioritisation of Functional Requirements and Non-functional Requirements in Agile Software Development

By

Ooi Jun Hao

Department of Information Systems and Analytics

School of Computing

National University of Singapore

2022/2023

B.Comp. Dissertation

Prioritisation of Functional Requirements and Non-functional Requirements in Agile Software Development

By

Ooi Jun Hao

Department of Information Systems and Analytics

School of Computing

National University of Singapore

2022/2023

Project No: H175480

Advisor: Prof Hahn Jungpil

Deliverables:

Report: 1 Volume

Abstract

Agile software development (ASD) has become a common practice that is used by many software development teams. However, due to the nature of being user centric and having a delivery oriented nature, teams often focus on fulfilling the functional requirements of a system and often neglect non-functional requirements. With common non-functional requirements such as performance, maintainability and security often overlooked, software systems quickly build up technical debt. With this tradeoff of having to balance the need to address non-functional requirements to prevent the overly quick build up of technical debt and the need to push out functional software requirements in a timely manner, we explore the question of prioritisation when it comes to functional requirements and non-functional requirements in ASD. Modelling the ASD process using *NK* fitness landscapes, we explore the differences in ASD project outcomes when placing varying levels of priority on non-functional requirements, alongside studying when would be the best point in the ASD cycle to introduce this focus on non-functional requirements.

Subject Descriptors:

D.2.1: Requirements/Specifications
I.6.4: Model Validation and Analysis
I.6.5: Model Development
K.6.3: Software Management

Keywords:

Non-functional requirements, agile software development, requirements prioritisation, *NK* fitness landscape model

Implementation Software and Hardware:

Python 3.7.15, Google Colaboratory, SoC Compute Cluster

Acknowledgements

The author wishes to express sincere appreciation to Professor Hahn and the members of the Garbage Can Lab at the NUS School of Computing for constructive feedback and suggestions on the research.

Table of Contents

1. Introduction

With business and technology requirements changing at an unprecedented rate, agile software development (ASD) has become widely adopted in many organisations. With a heavier focus on lean processes and dynamic adaptation as opposed to the traditional plan driven and structured approach of systems development (Nerur and Balijepally 2007), ASD gives rise to user centric and delivery oriented software development.

However, due to these characteristics of ASD, software projects are also prone to the build up of technical debt (Behutiye et al. 2017). Technical debt can be formally referred to as the trade off between expedient short-term decisions and the resulting, potentially crippling, long-term costs (Lim et al. 2012). To meet the demands of businesses, software teams that face challenges are inclined to take shortcuts to rapidly deliver functionality that is required, which has to be addressed in the future (Ramasubbu and Kemerer 2015).

With negative effects such as poor performance and poor quality (Rios et al. 2019), technical debt build up in ASD is partly due to lack of focus on non-functional requirements (NFR), where teams are overly-focused on the incremental delivery of functional features (Rios et al. 2019, Jarzębowicz and Weichbroth 2021). As defined in the IEEE software engineering standard 830-1998 (1998), non-functional requirements describe how a software system will provide the means to perform functional tasks. With common non-functional requirements such as performance, maintainability and security often overlooked in favour of functional requirements (FR) (Lim et al. 2012), software teams only take these into consideration when the need arises due to issues (Nguyen 2009). As the demand for software and complexity of software increases, technical debt build up and non-functional requirements should no longer be treated as a secondary objective, but rather a primary objective alongside the race to complete functional requirements (Rao and Gopichand 2011).

In this study, we explore the question of implementation of non-functional requirements alongside the implementation of functional requirements in ASD. Following the intuition that non-functional requirements should be a primary focus alongside functional requirements (Rao and Gopichand 2011), we look into the benefits and implications, as well as possible tradeoffs of working with both functional and non-functional requirements at the same time.

1.1. Literature Review

When looking at non-functional requirements and the way they are treated across different teams and organisations, Jarzębowicz and Weichbroth (2021) conducted a qualitative study on non-functional requirements in ASD to find out practices that are employed in industrial ASD projects to identify, elicit and document non-functional requirements. With a comprehensive review on non-functional requirements in regards to their role within the requirements engineering process, Jarzębowicz and Weichbroth (2021) concluded that elicitation and implementation techniques for non-functional requirements varied between ASD practitioners, each having a variety of approaches that yield satisfactory results.

Loucopoulos et al. (2013) argues that given the changing landscape in demand, the focus on non-functional requirements has become more important. He and his team then go on to propose a classification method to help in the classification of non-functional requirements followed by an analysis of existing classification methods. However, while Loucopoulos and his team came up with a very comprehensive classification method, there is no focus on how these classified non-functional requirements will be incorporated into agile sprints.

Amorndettawin and Senivongse (2019) proposed a strategy that involves taking non-functional requirements and mapping them into functional requirement templates. The goal of this would be to facilitate non-functional requirements gathering and to help agile team members push out the implementation of non-functional requirements within agile sprint cycles. On a more technical level, Odeh and Kamm (2003) reasoned that traditional software specification techniques such as the Unified Modelling Language (UML) are not enough to capture the complexity of organisational activities, which leads to missing focus on non-functional requirements when relying on such specifications to generate software requirements. Supakkul and Chung (2005) have tried to improve on this, and proposed an approach to take into account non-functional requirements when coming up with use-case diagrams via extensions of use case bubbles to show the non-functional requirements that are associated with a particular use case.

With a more direct focus on non-functional requirements prioritisation in ASD, Maiti and Mitropoulos (2017) proposed a mathematical approach in doing so. As part of a methodology called the Capture Elicit and Prioritising methodology, Maiti and Mitropoulos built on an

existing framework used for prioritising functional requirements onto non-functional requirements. Even though the study looked at the prioritisation of non-functional requirements in the beginning of agile software process, it is mainly focused on which non-functional requirement to prioritise as opposed to exploring whether or not to put focus on non-functional requirements alongside functional requirements (i.e., looking at whether to work on a non-functional requirement versus another non-functional requirement and not whether to focus on a non-functional requirement over a functional requirement).

Due to the critical nature of non-functional requirements and their ambiguity in comparison to functional requirements, much of the extant academic literature studying non-functional requirements is focused on defining, classifying and implementation strategies for non-functional requirements. Even though this lack of focus on non-functional requirements over functional requirements has been identified in much existing research, there has not been much study into how this issue can and should be resolved. Therefore, we put forward the following three research questions as seen in Table 1.

Question 1.	Should we always work on non-functional requirements alongside functional requirements?
Question 2.	How much focus should we put on non-functional requirements and functional requirements within each agile sprint iteration?
Question 3.	At what point in the project cycle should we introduce non-functional requirements?

Table 1. Summary of Research Questions

2. Computational Modelling

To explore the prioritisation of non-functional requirements in ASD and its implications, we set up a computational model to simulate the difference in outcomes when non-functional requirements are focused on early in the project development cycle as opposed to towards the end after all functional requirements have been catered to. Among the various simulation model approaches that are commonly used today, we employ the *NK* fitness landscape model

(Kauffman 1993, Kauffman and Weinberger 1989, Levinthal 1997) and extend it to the ASD process while considering the interdependencies between functional and non-functional requirements.

The use of the *NK* fitness model for this study is motivated by the model's ability to model interdependencies between requirements. Taking into consideration the unique interdependencies between functional and non-functional requirements, we can statistically explore the implications of taking on different prioritisation strategies (i.e., working on non-functional requirements alongside functional requirements or working on them separately). With systems development akin to being an iterative and incremental design problem solving process (Hahn and Lee 2021), the search for the highest value point in the *NK* fitness landscape is appropriate for our study in finding the optimum outcome based on different prioritisation strategies. To add on, the *NK* fitness model has been widely used in various organisational and management studies (Rivkin and Siggelkow 2003, Ethiraj and Levinthal 2009), signifying the confidence in the results produced from simulations based on the model. Our model will be used to explore the performance implications of the problem solving processes (i.e., ASD processes) of goal-directed adaptive agents (i.e., ASD team) when faced with different problem environments (i.e., FRs and NFRs).

2.1. *NK* Fitness Landscape Model

When working with a basic *NK* fitness landscape model, the researcher has to first specify the landscape configuration in which the problem space is being represented. In the context of our study, this would be an ASD project, **p**, which is represented by an N element vector of decisions (i.e., $\mathbf{p}=<d_1, d_2, \ldots, d_N>$), where each element can take a value of 0 or 1. Each decision, d_i, would have a contribution of c_i to the overall fitness of the project, $F(\mathbf{p})$. However, the value of c_i does not only depend on the choice of d_i, but also K other elements that are interdependent with d_i. This interdependency among decision variables is commonly represented in an NxN influence matrix.

$$\begin{array}{c} \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \end{array} \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Figure 1. Sample 6x6 Influence Matrix

In Figure 1, we can see a sample 6x6 influence matrix (i.e., N=6) with K=2, where a 1 on column j and row i signifies that the decision variable d_j influences the value of c_i, or 0 if otherwise. To illustrate this, using the example provided in Figure 1, we can see that $c_4 = c_4(d_4 \mid d_2, d_6)$.

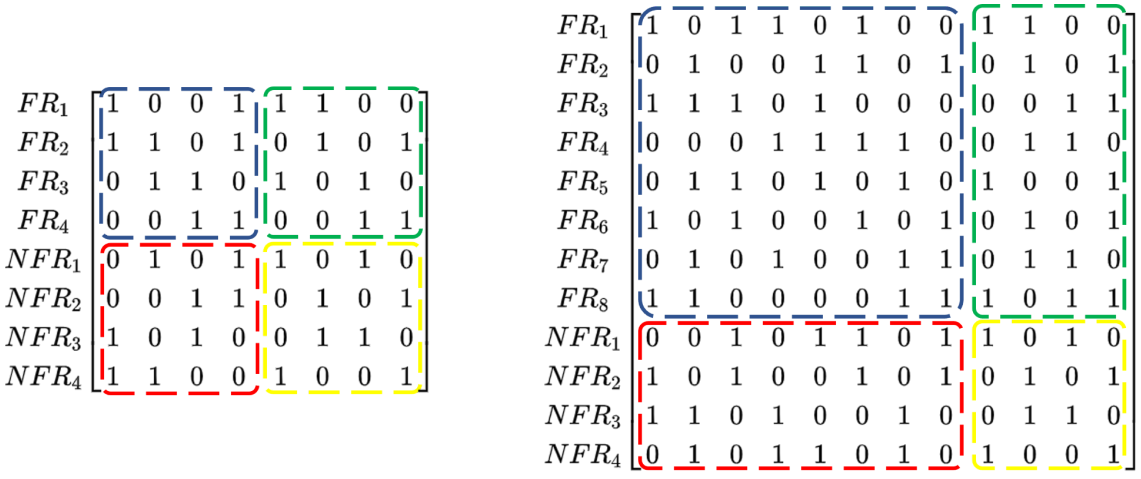
With the influence matrix specifying the interdependencies between decisions, we then draw random values of c_i (typically from a uniform distribution between 0 and 1) for each possible realisation of d_i and all its dependent decisions. Finally, we compute the overall fitness value of the configuration, $F(\mathbf{p})$, by taking the average over all the decision level fitness contributions. In essence, a higher $F(\mathbf{p})$ would signify a better performance and is more desirable.

The researcher would then have to specify an agent behaviour, where its behavioural rules (i.e., search method, etc.) are based on sound theories that model a real behaviour or a phenomenon of interest. For example, in the study on the evolution of information systems architecture by Haki et al. (2020), the agents are defined as actors (i.e., groups of humans) and IT applications (i.e., artefacts) who have a predefined set of attributes and behaviour that model the phenomenons of interest. In our case, an agent would be akin to an ASD team, where its adaptive behaviour is modelled as an incremental experiential search with differences in search strategies (more details on how this difference is modelled will be in a later section). For a generic agent behaviour, the search procedure would be, at each time period t, to select a decision choice at random and flip it (i.e., 1 to 0 or 0 to 1). We will then look at the new overall fitness score of this new configuration and compare it with the fitness score of the previous configuration (i.e., $F(\mathbf{p}_t)$ vs $F(\mathbf{p}_{t+1})$). If $F(\mathbf{p}_t)<F(\mathbf{p}_{t+1})$, we will deem the new configuration as a better configuration and adopt it moving forward, else we would keep

the old configuration. This process would repeat until we can no longer find a fitness improving alternative.

For the purpose of generalisation, we will configure our landscape space to follow requirement categories as opposed to specific requirements. This would mean that when mapping our requirement interdependencies and setting up the influence matrix for our model, we would be considering requirements based on their categories rather than specific user stories or non-functional requirements. Following the list of high level functional and non-functional requirements provided by the World Health Organisation (WHO) in their Digital Adaption Kit for Antenatal Care (2021), an example of a functional requirement would be that of being able to validate client details, whereas a non-functional requirement would be interoperability.[1]

A set up of an 8x8 and 16x16 influence matrix with 4 non-functional requirements each can be seen below in Figure 2. Since we are concerned between the unique interdependency of functional requirements with non-functional requirements, there are four quadrants that are of interest (listed from left to right, top to bottom in Figure 2) – 1) FRs dependent on other FRs, 2) FRs dependent on NFRs, 3) NFRs dependent on FRs, and 4) NFRs dependent on other NFRs. We will go into more details about these four quadrants and their configurations in the following sections.



$N = 8, Count(FR) = 4, Count(NFR) = 4$ $N = 12, Count(FR) = 8, Count(NFR) = 4$

Figure 2. Sample Influence Matrices for an ASD Project – FR and NFR

[1] Sample list of functional and non-functional requirements and its details can be found in Appendix B.

As for the prioritisation strategy, we will mainly be focusing on two methods of prioritising non-functional requirements, the first being the concurrent focus on non-functional requirements together with functional requirements (hereon referred to as **"together"**), and the second being the sole focus on functional requirements before moving on to non-functional requirements (hereon referred to as **"separate"**), which is the prevalent strategy currently adopted in practice.

To model this difference in strategy into the agent behaviour, differences will be made in the scope and sequencing of the search. Following the example seen in Figure 2 for N=8, when adopting the "separate" strategy, we will only focus on the first 4 decision choices (i.e., FR_1 to FR_4). Once a fitness improving alternative state can no longer be found, the search in the FR space is deemed as complete and we will move on to focus on the last 4 decision choices (i.e., NFR_1 to NFR_4). As for the "together" strategy, we will alternate between searching the FR space and the NFR space until we obtain the maximal performance.

For all experiments, we employ the Monte Carlo simulation method by running the simulation over 100 stochastically generated landscapes using the same problem structure, with 1,000 agents for each agent behaviour that are seeded onto the landscape with random initial configurations.

2.1.1. FR-NFR Interdependency

In the second and third quadrants, we are looking at the functional requirements' dependence on non-functional requirements and vice versa, respectively. Establishing the interdependency between functional and non-functional requirements is not as straightforward as just saying that all functional requirements are dependent on non-functional requirements and vice versa. This is ultimately due to the ambiguity in the way non-functional requirements are dealt with across organisations and across various methodologies. This difficulty in eliciting the interdependencies between functional and non-functional requirements has been addressed by Aguilar et al. (2012) in his study of requirement dependencies for managing and maintaining web applications, where he defines an algorithm to aid developers in eliciting the interdependencies between functional and non-functional requirements. Through the examples given in the study, where Aguilar et al.

(2012) defines dependencies to be either positive or negative, we can ultimately see that there is no fixed definition of interdependencies between functional and non-functional requirements.

Following the case study provided by WHO (2021), we can observe that for each functional requirement, there are some non-functional requirements that it depends on and there are some non-functional requirements that it does not depend on. Similarly, for each non-functional requirement, there are certain functional requirements that it depends on and certain functional requirements where it does not depend on. A proposal of a subset of the possible interdependencies can be seen in Figure 3.

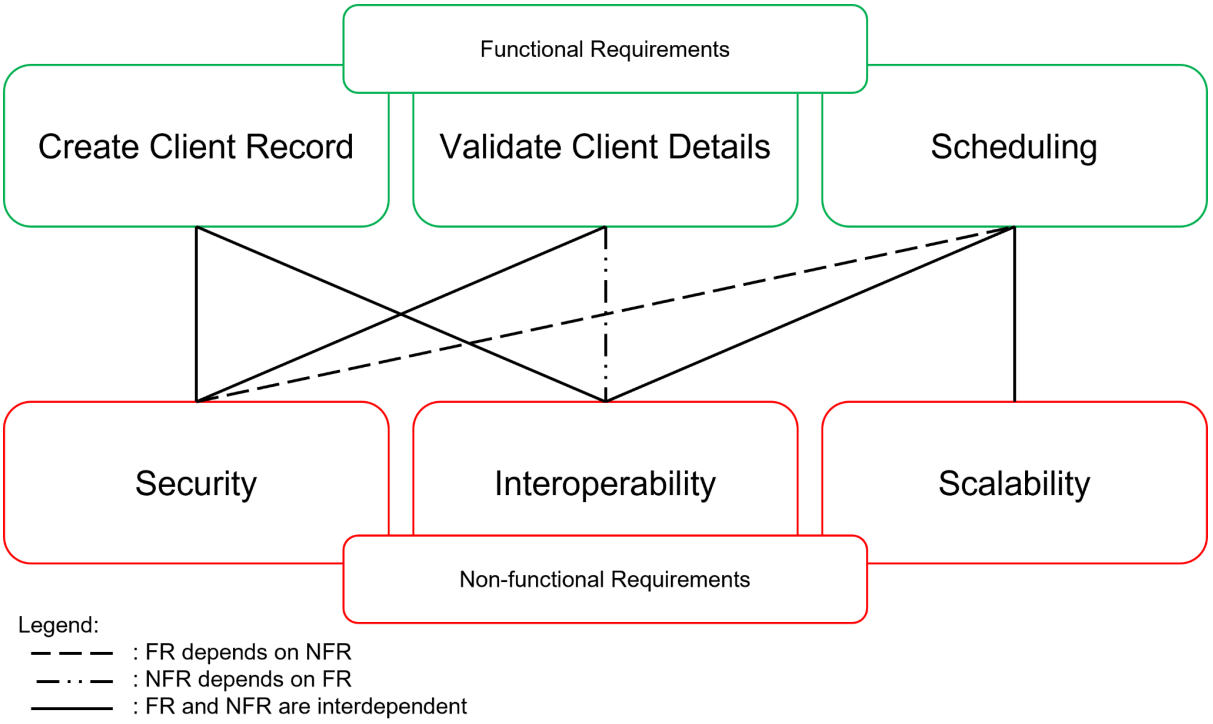


Figure 3. Proposed FR-NFR Interdependencies

From the proposed interdependencies in Figure 3, we can observe that the functional requirement of "validate client details" is dependent on the security aspect of the overall system as the system needs to securely exchange and verify client information. However, it is not dependent on the scalability aspect of the entire system as whether or not the system is designed to be scalable has nothing to do with being able to validate a client's details on the system. On the other hand, whether or not the system is secure, does not depend on how the "scheduling" function is implemented but rather on how the "create client record" and

"validate client details" functions are implemented as they involve sensitive information that needs to be protected from unauthorised access or manipulation. This ultimately goes to show the possibility of varying levels of interdependencies between functional and non-functional requirements.

The interdependencies between functional and non-functional requirements will vary greatly from project to project and might even be subjective to ASD teams depending on how the non-functional requirements are defined. Therefore, it would not be possible to make a simple generalisation on their interdependencies and we will experiment with different combinations of interdependencies between functional and non-functional requirements. An illustration of how these differences will be captured in the landscape's influence matrix can be seen in Figure 4.

Figure 4. Sample FR-NFR Interdependency Influence Matrices

2.1.2. NFR Interdependency

When taking a look at the interdependencies between non-functional requirements (i.e., the fourth quadrant), non-functional requirements often tend to conflict with one another (Dewi et al. 2009). Using the case study provided by WHO (2021) as an illustration, we cannot rule out the possibility that the non-functional requirements of scalability and security might conflict with one another (i.e., when the security of the system is improved, it might make it more difficult to scale the system up to meet greater user traffic due to bottlenecks posed by security aspects of the system).

Even though it is known that non-functional requirements tend to interfere and depend on one another, it is difficult to make a generalisation on their interdependencies due to the ambiguous nature of non-functional requirements. With the inherent difficulty in managing the variations of interdependencies between non-functional requirements, Tabassum et al. (2014) has devised a framework that helps illicit and deal with interdependent non-functional

requirements that, reducing the chance of conflict between non-functional requirements and to improving the overall management of the software.

Similarly, we will experiment with different combinations of interdependencies between non-functional requirements. An illustration of how these differences will be captured in the landscape's influence matrix can be seen below in Figure 5. One difference to note here as compared to the experiments conducted previously on FR-NFR interdependencies would be the possibility that there are no interdependencies between the functional requirements (i.e., complete separation of concerns).

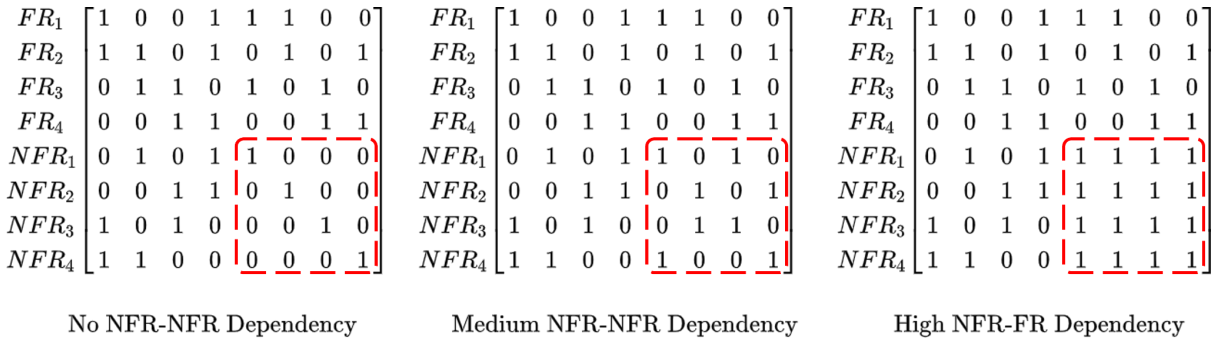


Figure 5. Sample NFR Interdependency Influence Matrices

2.1.3. FR Interdependency

When looking at the interdependencies between functional requirements (i.e., the first quadrant), we follow the notion that the extent of interdependencies affects the complexity of the overall project, with a higher level of dependency resulting in a higher level of complexity (Xia and Lee 2005). It is also worth noting that it is not just the level of interdependency between requirements that determines how complex a system is, but also the pattern of interdependencies determines the complexity of the system (Rivkin and Siggelkow 2007).

Dealing with complexity in software development is a topic that has been widely reviewed in existing literature, with consensus that a higher level of complexity results in a lower level of user satisfaction and system functionality (Xia and Lee 2004). As such, we will be varying the level of complexity in our simulations to study the potential difference in outcomes based on non-functional requirement prioritisation strategies. An illustration of how this difference in complexity of the ASD project is illustrated in Figure 6.

FR_1	1	0	0	0	1	1	0	0
FR_2	0	1	0	0	0	1	0	1
FR_3	0	0	1	0	1	0	1	0
FR_4	0	0	0	1	0	0	1	1
NFR_1	0	1	0	0	1	0	1	0
NFR_2	0	0	0	1	0	1	0	1
NFR_3	1	0	0	0	0	1	1	0
NFR_4	1	0	0	0	1	0	0	1

Low Project Complexity

FR_1	1	1	1	1	1	1	0	0
FR_2	1	1	1	1	0	1	0	1
FR_3	1	1	1	1	1	0	1	0
FR_4	1	1	1	1	0	0	1	1
NFR_1	0	1	0	0	1	0	1	0
NFR_2	0	0	0	1	0	1	0	1
NFR_3	1	0	0	0	0	1	1	0
NFR_4	1	0	0	0	1	0	0	1

High Project Complexity

Figure 6. Sample Project Complexity Influence Matrices

2.2. Baseline Validation

In any study using simulation methods, the validation of the computational model is an essential step in any theory development (Davis et al. 2007). To validate our model, we will conduct exhaustive tests across the two prioritisation strategies on all the possibilities of interdependencies for each quadrant as discussed previously. For our simulations, we configure the landscape to contain a total of 12 requirement decisions (i.e., N=12), consisting of 8 functional requirements and 4 non-functional requirements.

Starting off with the FR-NFR dependencies (i.e., functional requirements that depend on non-functional requirements), we experiment with varying levels of dependencies and run the simulation over the two different prioritisation strategies. Figure 7 shows the results when we vary the level of functional to non-functional requirements dependencies (i.e., second quadrant).

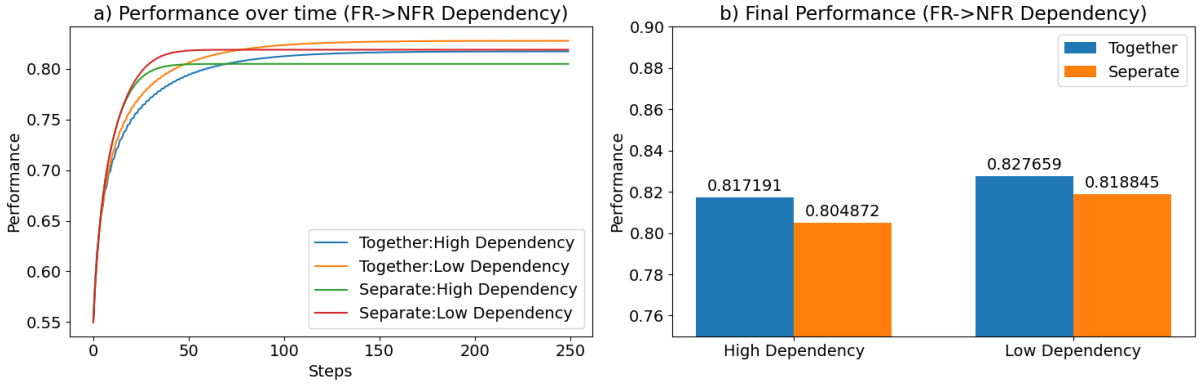


Figure 7. FR->NFR Dependency Results

Regardless of the level of dependencies of functional requirements on non-functional requirements, the performance score achieved is always higher when we adopt the “together” strategy over the “separate” strategy. Furthermore, we can see that regardless of the level of dependencies (i.e., high dependency vs low dependency), the difference in performance is similar. This would imply that it is equally beneficial to work on functional requirements alongside non-functional requirements whether or not functional requirements have a high level of dependency on non-functional requirements.

We then experiment with varying levels of NFR-FR dependencies across the two prioritisation strategies (i.e., non-functional requirements that depend on functional requirements). Figure 8 shows the results when varying the level of non-functional to functional requirements dependencies (i.e., third quadrant).

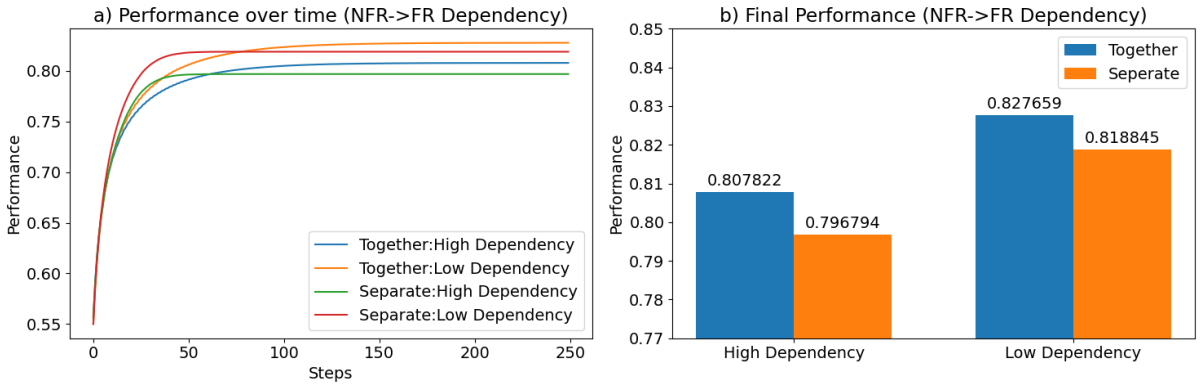


Figure 8. NFR->FR Dependency Results

Similar to the results we achieved previously, the performance score achieved is always higher when we adopt the “together” strategy over the “separate” strategy. Furthermore, we can see that regardless of the level of dependencies, the difference in performance is similar. This implies the equal benefit of utilising the “together” strategy over the “separate” strategy regardless of dependency levels.

Moving on, we experiment over varying levels of NFR interdependencies (i.e., non-functional requirements depending on other non-functional requirements). The results for the simulations over varying levels of NFR interdependencies can be seen in Figure 9.

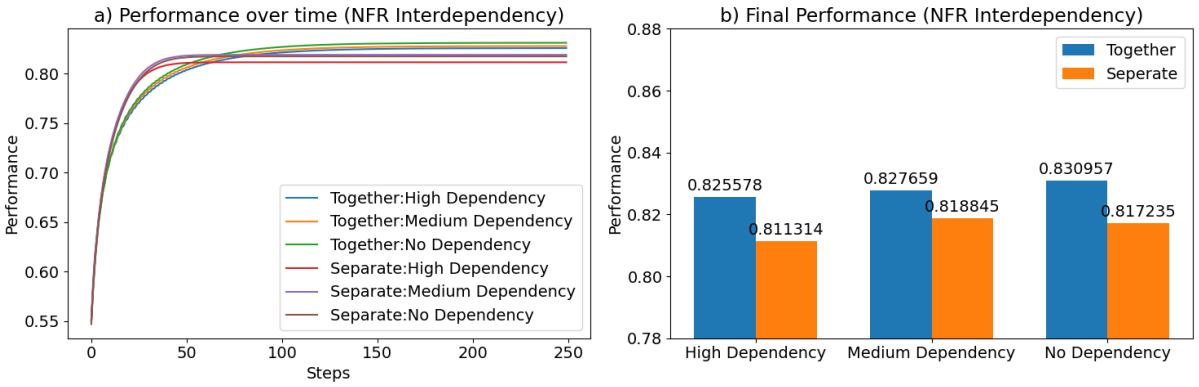


Figure 9. NFR Interdependency Results

Regardless of the level of interdependency between non-functional requirements, the final performance score obtained when doing the "together" strategy is greater when compared to doing the "separate" strategy. On top of that, the difference in final performance score over all levels of interdependencies are similar. This again implies that regardless of the extent of interdependency between non-functional requirements, the benefit obtained when working on non-functional requirements alongside functional requirements is the same.

Last but not least, we experiment with varying levels of FR interdependencies (i.e., functional requirements depending on other functional requirements), where a higher level of interdependencies signifies a higher level of overall project complexity. In addition, we also vary the number of functional requirements (i.e., project size). The results are summarised in Figure 10 and 11.

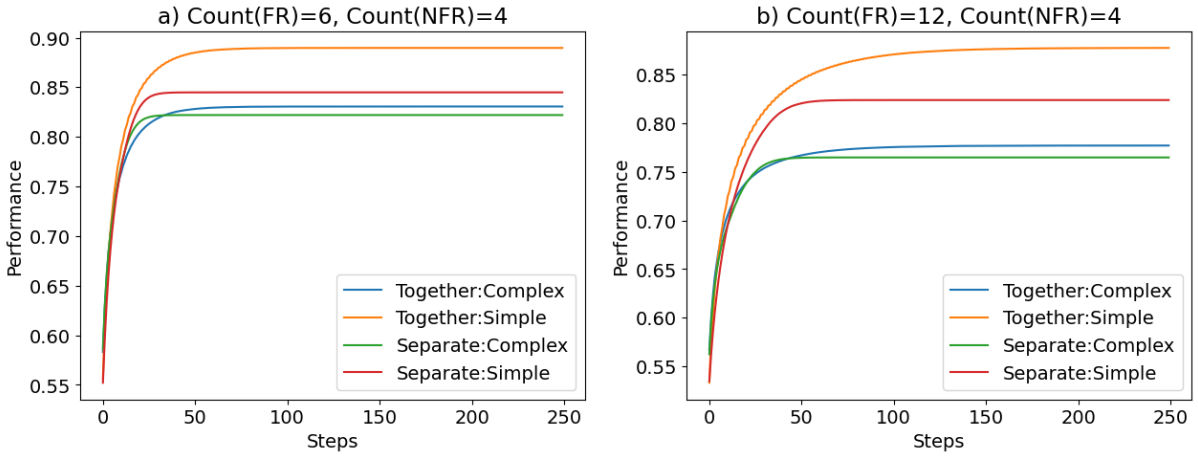


Figure 10. FR Interdependency Performance over time

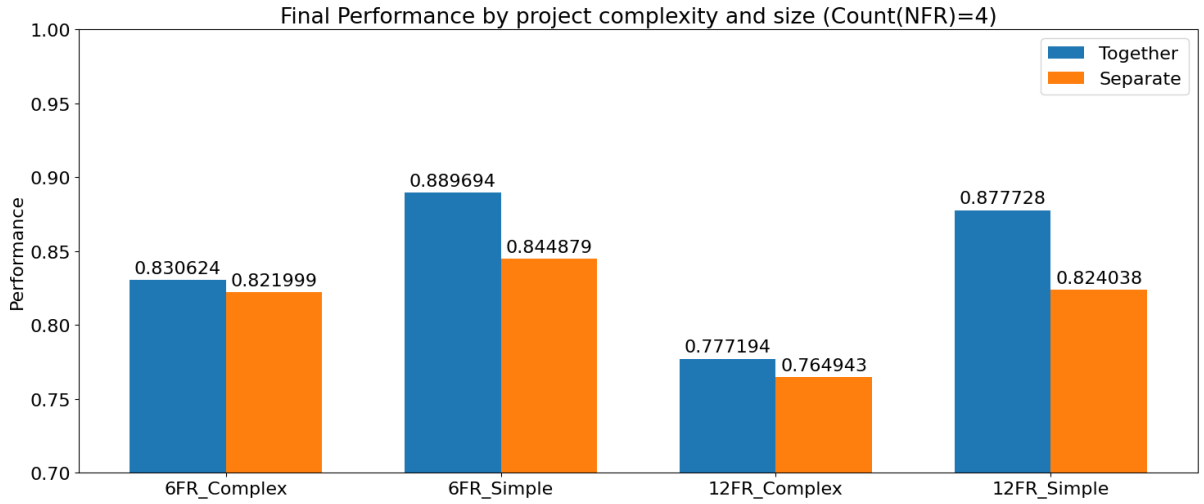


Figure 11. FR Interdependency Final Performance

We can see that regardless of project size or complexity, the optimum performance score obtained when carrying out the "together" strategy is better than the "separate" strategy. This would suggest that regardless of the size of the project and the complexity of the functional requirements, an ASD team should always work on non-functional requirements alongside functional requirements, instead of leaving non-functional requirements towards the end. Additionally, we can observe that the difference in performance score is greater when the project is simple as compared to when the project is complex regardless of project size, implying that it is more beneficial to work on non-functional requirements alongside functional requirements when there is a lower level of dependencies between functional requirements (i.e., project has a low complexity level).

Through our simulations, we are able to validate our *NK* model against existing research findings and our intuition. In particular, we are able to observe that final performance decreases as project complexity increases, which supports the notion that a higher level of complexity results in a lower level of user satisfaction and system functionality (Xia and Lee 2004). Additionally, we are able to observe that working on non-functional requirements alongside functional requirements results in a higher final performance when compared to working on only functional requirements in the beginning, which supports our intuition given accounts of quicker technical debt buildup due to negligence on non-functional requirements (Rios et al. 2019, Jarzębowicz and Weichbroth 2021).

Given the equal increase in performance when adopting the "together" strategy over the "separate" strategy for all levels of interdependencies between functional and non-functional requirements as well as between non-functional requirements, we will thus adopt a generalised approach moving forward on the interdependencies between functional and non-functional requirements and between non-functional requirements for brevity (i.e., medium level of interdependency for FR->NFR, NFR->FR, and NFR->NFR).

3. Prioritisation of NFRs and FRs

With the validation of our baseline computational model, we extend our model and the simulations to address the three main research questions of our study. To reiterate, we are interested in exploring the following questions – 1) Is working on non-functional requirements alongside functional requirements always more beneficial, 2) How much focus should we put on non-functional requirements over functional requirements in each iteration, and 3) When should we introduce non-functional requirements into the scope of requirements to fulfil.

3.1. Extended *NK* Fitness Landscape Models

We simulate various conditions and extensions to the model that emulate scenarios that have been identified in existing literature to test the robustness of the results obtained from our baseline model, as well as to test if working on non-functional requirements and functional requirements together is always better than doing them separately. For this, we will experiment with 4 different sets of conditions – 1) Lack of Knowledge, 2) Multiple Goals, 3) Project Modularity, and 4) Resource Availability. For all experiments, we conduct the simulation on a landscape containing 12 requirement decisions (i.e., N=12), consisting of 8 functional requirements and 4 non-functional requirements.

3.1.1. Lack of Knowledge

In our baseline experiments conducted above, we are working on the underlying assumption that the ASD project team is aware of the existence and implications of non-functional requirements. However, more often than not, development teams overlook non-functional requirements to the extent that they are not even identified in the requirements elicitation process (Amorndettawin and Senivongse 2019). This would create implications when we

conduct a search using the "separate" strategy. Since the significance and influence of non-functional requirements is considered to be overlooked by the team in this strategy, the team would not be aware of the implications of the non-functional requirements on the functional requirements until later in the development life cycle.

To model this behaviour, we adopt a similar approach as Hahn and Lee (2021) in their study of cross domain knowledge on information systems development. Simply put, since the team is not aware of the dependence on non-functional requirements, any step in searching through the functional requirement space would produce a fitness contribution of the average of all possible combinations of non-functional requirements. An illustration of how this would be calculated can be seen below in Table 2. The results of incorporating lack of knowledge is shown in Figure 12.

Fitness Contribution for "Separate" Strategy	
$\mathbf{p} = < d_{FR1}, d_{FR2}, d_{FR3}, d_{FR4}, d_{NFR1}, d_{NFR2} >$	
Configuration	**Fitness Contribution of FR_1**
< <u>0</u>, 0, 1, 1, **0, 0** >	0.912
< <u>0</u>, 0, 1, 1, **0, 1** >	0.731
< <u>0</u>, 0, 1, 1, **1, 0** >	0.422
< <u>0</u>, 0, 1, 1, **1, 1** >	0.697
< <u>0</u>, 0, 1, 1, **?, ?** >	0.6905 (average)

Table 2: Numerical Example of Fitness Contribution when tuning d_{FR1}

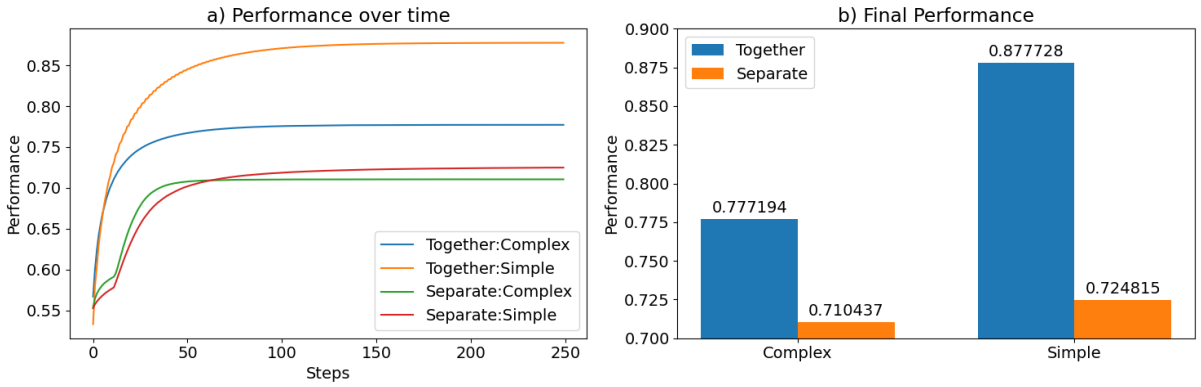


Figure 12. Lack of Knowledge Landscape Results

Looking at the results in Figure 12, we can conclude that regardless of complexity, the optimum fitness score obtained when carrying out the "together" strategy is better than the "separate" strategy (similar to the results obtained from our baseline validation). One interesting observation to note would be the difference in performances between strategies when using this landscape as opposed to the baseline landscape. We can see that the difference in performance values here are larger across the varying levels of complexity. This would imply that the benefit of adopting the "together" strategy is actually much greater than what we initially concluded in our baseline landscape. Furthermore, similar to our baseline results, we can see that the benefit from adopting the "together" strategy over the "separate" strategy is greater when the project is simple as compared to when the project is complex due to the greater difference in final performance obtained.

3.1.2. Multiple Goals

In the study on managing non-functional requirements in practice carried out by Werner et al. (2022), organisations in practice have been able to realise certain non-functional requirements of a system (i.e., availability, scalability, etc.) by employing services from third party vendors, such as cloud platforms and devops services. Though this process of offloading the responsibility of maintaining a set of non-functional requirements helps in reducing cost and saving time for the firm, firms often lose control and become overly dependent on the third party service provider (Werner et al. 2022). Ultimately, offloading the non-functional requirements to third parties that can help realise it and provide guaranteed quality of service (Anisetti 2020) comes with tradeoffs.

To study this potential effect of offloading the non-functional requirement to a third-party provider when considering the different prioritisation strategies, we try to adopt a goal oriented approach (i.e., goal of maximising functional performance vs goal of maximising non-functional performance). This approach of viewing the goal of maximising functional performance and the goal of maximising non-functional performance is adapted from Ethiraj and Levinthal (2009) in their research regarding complex organisations and multiple goals. With the realisation of non-functional requirements outsourced to third-parties, we can view it as two separate entities working within the same project (i.e., the ASD team and the third-party service provider), both looking to achieve their respective goals of maximising functional requirement performance and maximising non-functional requirement performance, respectively.

Using this method, we would consider the functional requirements and non-functional requirements as two separate landscapes, taking their corresponding fitness scores as the goals (i.e., the goal of maximising functional performance would be finding the best fitness score of the functional requirements landscape). In this case, when we adopt the "separate" approach, we would only care about the fitness score of the sub-landscape that we are working on with no regards for the other. Whereas for the "together" approach, we would search a landscape with consideration of the fitness score of both the landscape that we are searching on as well as the other landscape.

In this case, the overall fitness for the goal of maximising functional performance vs goal of maximising non-functional performance would be $F(\mathbf{p_{FR}})$ and $F(\mathbf{p_{NFR}})$, respectively, where $\mathbf{p_{FR}}=<d_{FR1}, d_{FR2}, \ldots, d_{FR7}, d_{FR8}>$ and $\mathbf{p_{NFR}}=<d_{NFR1}, d_{NFR2}, d_{NFR3}, d_{NFR4}>$. In the case of the "separate" strategy, where we first focus on the functional requirements before moving on to the non-functional requirements, we will deem an alternate state to have a better fitness if $F(\mathbf{p_{FR,\,t+1}}) > F(\mathbf{p_{FR,\,t}})$ when focusing on the functional requirements, else $F(\mathbf{p_{NFR,\,t+1}}) > F(\mathbf{p_{NFR,\,t}})$ when focusing on the non-functional requirements. Whereas in the case of the "together" strategy, we will deem an alternate state to have a better fitness if $F(\mathbf{p_{FR,\,t+1}}) > F(\mathbf{p_{FR,\,t}})$ and $F(\mathbf{p_{NFR,\,t+1}}) \geq F(\mathbf{p_{NFR,\,t}})$ when focusing on the functional requirements, else $F(\mathbf{p_{NFR,\,t+1}}) > F(\mathbf{p_{NFR,\,t}})$ and $F(\mathbf{p_{FR,\,t+1}}) \geq F(\mathbf{p_{FR,\,t}})$ when focusing on the non-functional requirements. This rule is derived from the intuition that when doing functional requirements and non-functional requirements separately in the "separately" strategy, we are only concerned with the goal that we are working on and nothing else, whereas if we are doing functional requirements and

non-functional requirements concurrently in the "together" method, we are keeping in mind both goals as we work on either one.

The final performance of the entire ASD project will be based on the average of the score of the goal of maximising functional performance and the score of the goal of maximising non-functional performance (i.e., both goals hold equal importance to the overall project). This would be represented by $F(\mathbf{p}) = (F(\mathbf{p}_{FR}) + F(\mathbf{p}_{NFR})) / 2$.

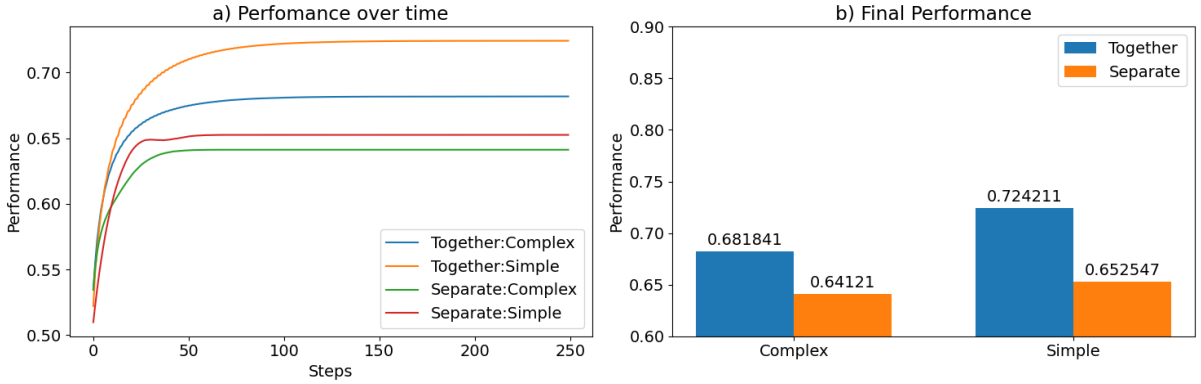


Figure 13. Multiple Goals Landscape Results

With the results of the experiment shown above in Figure 13, we can see that along the search process, the overall fitness score of the project actually decreases at certain parts of the graph for the "separate" strategy. This is ultimately due to the fact that whilst we are searching one landscape, we do not care about the decrease (or increase) of performance of the other landscape.

Similar to the conclusions drawn before, we can see that regardless of complexity, it is still better to take on the "together" approach as opposed to the "separate" approach. However, even though the difference in fitness scores between the two strategies is still greater when the project is simple, the gap in the differences are now smaller as compared to earlier experiments.

3.1.3. Project Modularity

Formally defined as a set of principles for managing complexity by breaking up a system into discrete portions that then communicate with one another through standardised interfaces within a standardised architecture (Langlois 2002), project modularity in the context of

software development has been widely studied in existing literature. With rising popular trends such as component based development and adoption of object oriented languages (i.e., practises that display the paradigm of modularity), the concept of modularity has become a common practice amongst teams in a bid to speed up and increase quality of software projects (Narduzzo and Rossi 2003).

In theory, software modularity affects the performance of an ASD project as the modular approach helps in breaking up tasks that are complex and interdependent, helping improve efficiency when organising and managing ASD projects (Yeo and Hahn 2014), which can ultimately improve the final result of the project. As such, we study the difference in non-functional requirement prioritisation strategy under different levels of project modularity to see if there are any differences in outcome.

Adapting a similar approach to the study of modularisation in information systems development conducted by Yeo and Hahn (2014), we arrange the interdependencies of the functional requirement decision variables into clusters to model the different modules within an ASD project. We will configure our landscape to follow two levels of project modularisation, a fully modularised landscape and a non-modular/random landscape amongst functional requirements (seen in Figure 14). The total number of interdependencies between functional requirement decision variables is kept constant across the two levels of project modularisation, ensuring the same overall level of complexity for the ASD project.

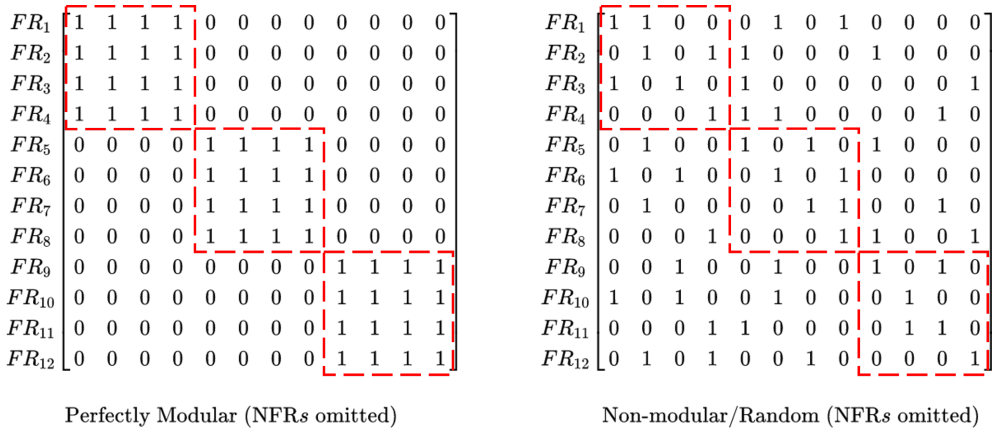


Figure 14. Sample Influence Matrices with varying Levels of Modularisation

A slight change will be made to the agent search behaviour as well. When adopting the "together" strategy of working on non-functional requirements alongside functional requirements, we will work within the modules that are defined in the scope of functional requirements before looking at the scope of non-functional requirements. Following the example in Figure 14, we will start off the search process by conducting the incremental search within decisions FR_1 to FR_4. Once an optimal configuration has been found within these 4 decisions, we will then move on to search the non-functional space (which is not shown in Figure 14 for brevity). Once the configurations have been exhausted within the non-functional space and the optimal fitness score has been achieved, we will then return to the functional requirements space and move on to search the next module (i.e., FR_5 to FR_8). This process would repeat itself until no fitness improving alternative can be achieved throughout the entire project. As for the "separate" strategy, we would conduct the search in the functional requirements space on a module basis, where we start off in the first module (i.e., FR_1 to FR_4) until no fitness improving alternative can be found before moving on to the second module (i.e., FR_5 to FR_8). This process would repeat itself until no more fitness improving alternatives can be found throughout the entire functional requirements space, in which we will then move on to the non-functional requirements space. The results for the simulation are shown in Figure 15.

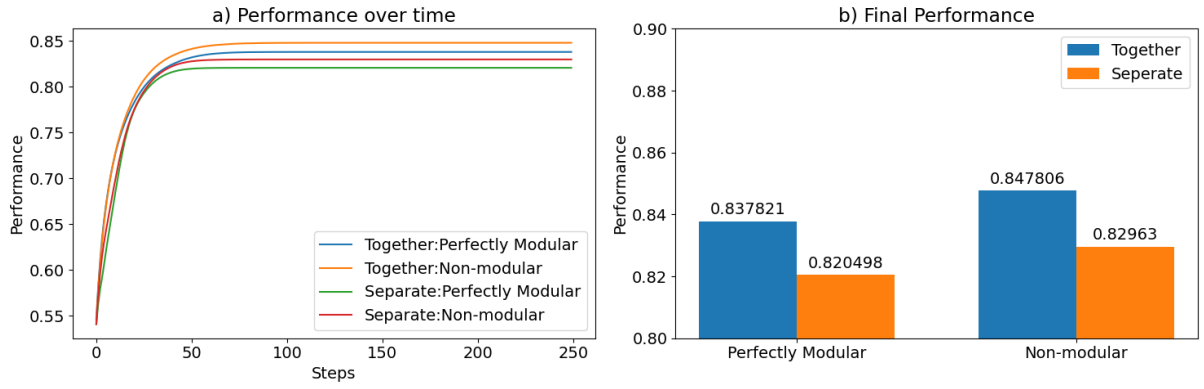


Figure 15. Project Modularity Results

From the results of the simulation, we can observe that the "together" strategy still yields a better performance as compared to the "separate" strategy regardless of the level of modularisation. When observing the differences in fitness scores obtained across the different levels of modularisation, we can see that the benefit gained from carrying out the "together" strategy is fairly similar. Furthermore, the results here are consistent with the results

presented by Yeo and Hahn (2014), where a non-modular project structure yields a better result as compared to a perfectly modular structure when carrying out the project in an agile manner.

3.1.4. Resource Availability

Last but not least, we consider the effect of the project resource availability while examining the difference between the two prioritisation strategies. Resource availability represents the constraint of limited cost, time and/or manpower that affects an ASD team's ability to work on the project to the best of their abilities. To model this constraint in our experiments, we will adopt a similar approach to Yeo and Hahn (2014) and limit the number of neighbouring decision variables that can be considered when doing our search on the landscape. This would mean that instead of performing an exhaustive search of alternative configurations, we would only be able to investigate a subset of them (e.g., 50% of all possible neighbouring configurations). Figure 16 summarises the results.

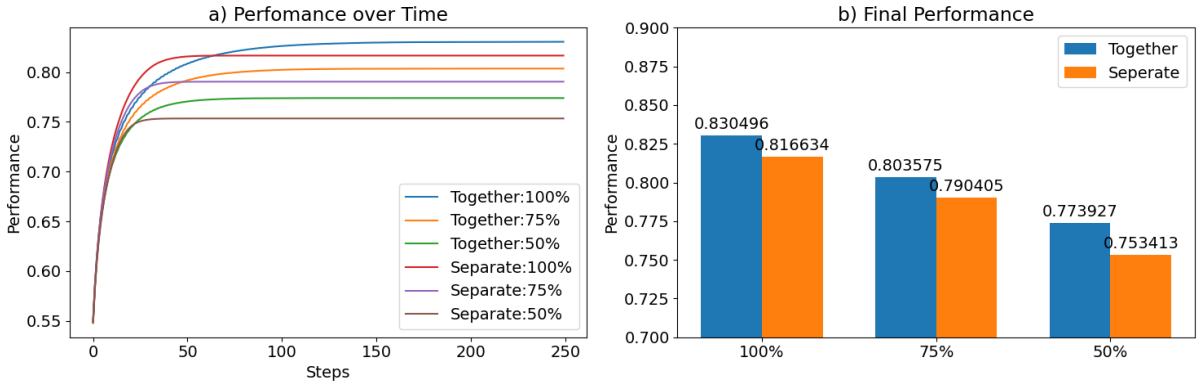


Figure 16. Resource Availability Results

We run the experiment over three levels of resource availability (i.e., 100%, 75% and 50%), where 100% would represent the absence of any type of constraint placed on the ASD team. As seen in Figure 16, regardless of the level of resource availability, the results are consistent with all prior experiments in that the "together" strategy yields a better result when compared to the "separate" strategy. When looking across the varying levels of resource availability, there does not seem to be a clear trend in the difference in benefit gained when the level of resource availability varies.

From the consistent result across the various extensions and scenarios, we are able to conclude that working on non-functional requirements alongside functional requirements will always yield a better outcome when compared to just focusing on functional requirements before moving on to non-functional requirements. Moving forward, on top of just working on non-functional requirements alongside functional requirements, we explore various aspects of how we would work on non-functional requirements alongside functional requirements.

3.2. Focus Level of NFR vs FR

With the conclusion that working on non-functional requirements alongside functional requirements is always more beneficial than working on the two separately, the question of how much focus developers should put on functional requirements compared to non-functional requirements still remains. That is to say, during agile sprints, what percentage of the time should be spent focusing on fulfilling the functional aspects of the system and what percentage of the time should be spent focusing on the non-functional aspects of the project. In our previous simulations using the "together" strategy, an equal focus on both functional and non-functional requirements has been placed due to the alternating search between the two spaces throughout the search process.

We will adjust the agent behaviour to study this issue using the *NK* fitness landscape setup. In our previous experiments, we tested the difference between two agent behaviours that followed the different prioritisation strategy of "together" and "separate". For this experiment, we will introduce a probability factor when it comes to the scope of the search. Over the two different search scopes (i.e., FR scope and NFR scope), the agent will now have a probability of x ($0 \leq x \leq 1$) to tune a random decision in the FR scope and a probability of $1\text{-}x$ to tune a random decision in the NFR scope. Following the example shown back in Figure 2 (where N=12), this would mean that for each iteration of the search process, the agent would have a probability of x to conduct the search on a random decision from FR_1 to FR_8, and a probability of $1\text{-}x$ to conduct the search on a random decision from NFR_1 to NFR_4. We run the experiment over various levels of x (i.e., FR Focus Level) and over various complexities and sizes of the project. The difference in project complexity and size will be captured using varying levels of interdependency between FRs and varying numbers of FRs, respectively. The full results of the difference in performance over time when varying the

level of focus on functional requirements over various levels of complexities can be seen in Figure 17.[2]

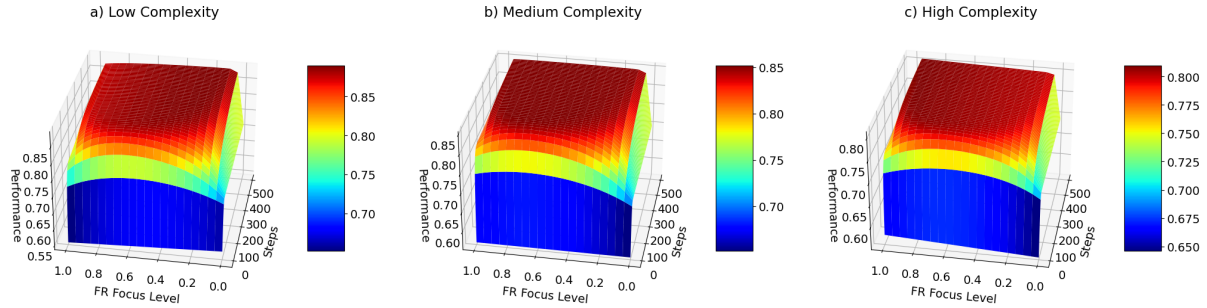


Figure 17. FR Focus Level Results

Over all levels of complexity, we can observe that the final fitness scores obtained are quite similar across all levels of FR focus, excluding the edge cases where we run the experiment with a 1 or 0 probability of focusing on FRs. However, we can see that the rate of change of performance (especially when approaching the convergence point) are different across the various levels of FR focus. This would imply that the time taken to hit the best fitness score varies across different levels of FR focus, which is akin to saying that the time spent on the ASD project to hit our desired state is different depending on the level of focus that we put into functional and non-functional requirements. To further aid in our analysis, we obtain the final performance and the convergence step for the different levels of complexities (seen in Figure 18).

[2] As the conclusions drawn are consistent across varying project sizes, the results for varying project sizes have been omitted for brevity in this report. See Appendix A for access to full results.

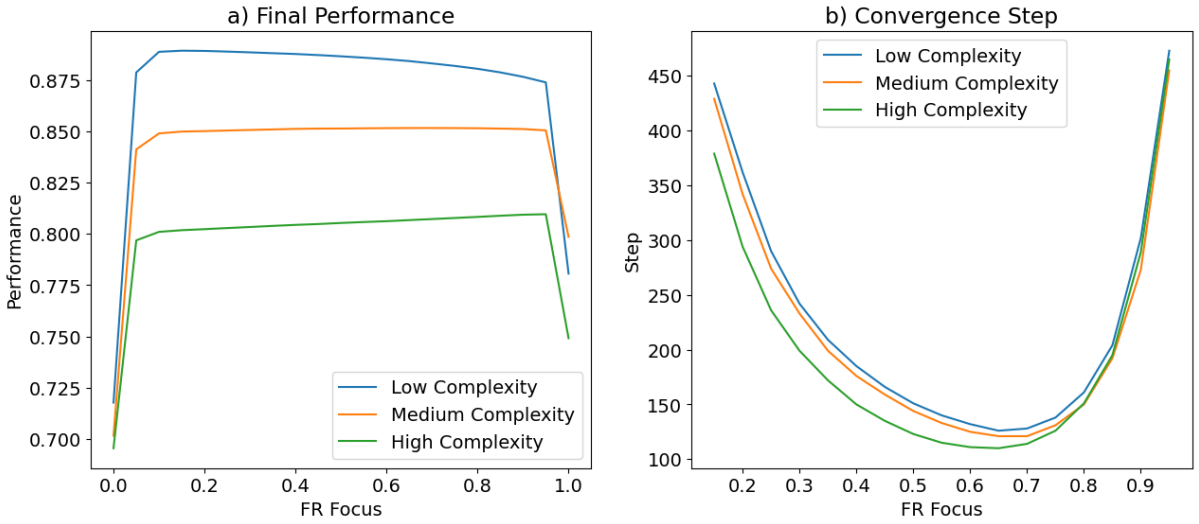


Figure 18. FR Focus Level – Final Performance and Convergence Step

From the results, we can observe that regardless of complexity, the optimal fitness is obtained the quickest when the FR focus level is set around the 0.5 to 0.8 range. This would imply that if we were to put a 50% to 80% focus on functional requirements and the rest of the time on non-functional requirements, we would be able to obtain the final result in the shortest amount of time. In the case of the final fitness score achieved, the performance results are relatively consistent across the different FR focus levels (excluding the edge cases where FR focus is 0 and 1). Additionally, there is no discernible difference in pattern across varying project complexity.

We are thus able to conclude that regardless of project complexity, we should put a moderate amount of focus on the functional over the non-functional aspects of the system without an overemphasis on it if we are looking to complete the project in the quickest time possible and achieve the optimum result. A caveat to this conclusion would be that this result would still be dependent on how we define non-functional requirements within ASD projects (i.e., how it is elicited, implemented, etc.). The way that the non-functional requirements are defined and implemented, would ultimately define what it means to be more focused on functional requirements over non-functional requirements and vice versa.

3.3. Introduction of NFRs

In addition to the focus levels between functional and non-functional requirements, the question of timing should also be considered. More often than not, non-functional

requirements are not elicited at the same time as functional requirements (Heumesser et al. 2003, Yusop et al. 2008) and are often treated superficially in early phases on the ASD project (Cao and Ramesh 2008). This indicates that it is perhaps not just the way non-functional requirements are defined and implemented, but it is also about the timing in which non-functional requirements are introduced to the team that could influence the outcome of the project. Currently, there is no consensus among practitioners on when the introduction of non-functional requirements into the project life cycle would be ideal (Jarzębowicz and Weichbroth 2021).

To study this question of timing, we will run our simulation with varying steps in which the search in the NFR space is introduced into the search process. For example, if the step to introduce NFRs into the search were to be set as *t*, then for the steps 0 to *t*-1, we would be only focusing on the FR space. From step *t* until the end of the search iteration, we will alternate between the FR space and the NFR space until we obtain the maximum performance (similar to the "together" strategy). We run the experiment over various levels of *t* (i.e., NFR introduction step) and over various complexities and sizes of the project. The difference in project complexity and size will be captured using varying levels of interdependency between FRs and varying numbers of FRs, respectively. The full results of the difference in performance over time when varying timing of the introduction of non-functional requirements over varying levels of project complexities can be seen in Figure 19.[3]

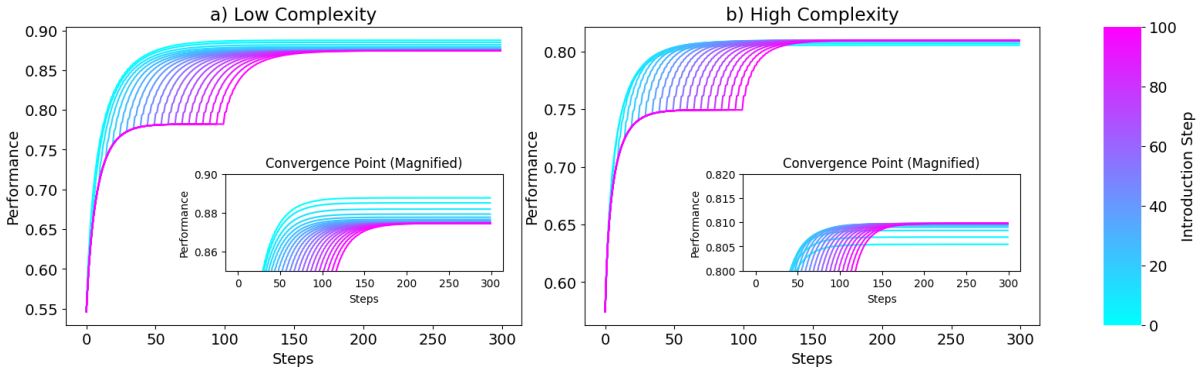


Figure 19. NFR Introduction Results

[3] As the conclusions drawn are consistent across varying project sizes, the results for varying project sizes have been omitted for brevity in this report. See Appendix A for access to full results.

With a focus on the final convergence point (i.e, the magnified portion), we notice that there is a difference in behaviour across the complexity levels when it comes to the timing in which the search on NFR is introduced. To study this behaviour, we thus plot the final performance against the NFR introduction step in Figure 20.

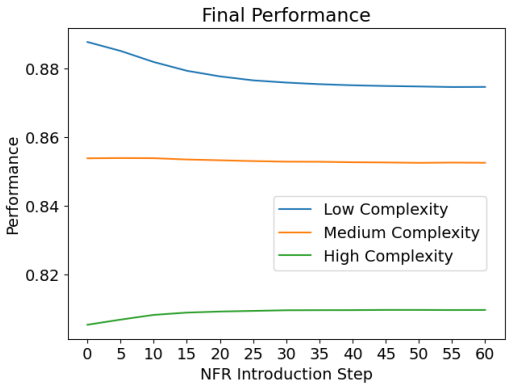


Figure 20. NFR Introduction – Final Performances

We can see that when project complexity is low, the performance decreases as we introduce NFR in a later step. However, when project complexity is high, an opposite pattern is observed, where a later step in which NFRs are being introduced leads to a better performance (up until a certain point where the performance converges). This would imply that when project complexity is low, implementation of non-functional requirements should be done alongside functional requirements right from the beginning. On the other hand when project complexity is high, implementation of non-functional requirements alongside functional requirements should begin some time after the project starts, where the initial phases of the project are focused mainly on functional requirements. On top of that, we can see that there is a slight difference in the gradient (i.e., rate of performance increase/decrease) when it comes to the varying levels of complexities. This implies the greater the influence of the timing in which NFRs are introduced on performance when complexity of the project is low as compared to when complexity of the project is high.

4. Conclusion

4.1. Summary

In this study, the ASD process is simulated using the *NK* fitness landscape model where each decision node represents a functional or non-functional requirement. Using a Monte Carlo simulation approach, we studied the differences in final performance obtained when we employed different agent behaviours that follow the different ways in which teams can deal with non-functional requirements in ASD. The propositions derived from this study are summarised in Table 3.

Proposition 1.	Working on non-functional requirements alongside functional requirements is always more beneficial than working on the two separately.
Proposition 2.	Regardless of project complexity, a moderate focus should be placed on fulfilling functional requirements over non-functional requirements without an overemphasis on functional requirements.
Proposition 3.	Non-functional requirements should be implemented right from the beginning of an ASD project and should be held off for a while before being introduced when the project is simple and complex respectively.

Table 3. Summary of Propositions

From our baseline model validation and the extensions that emulate real life accounts of software projects and the effect that non-functional requirements pose on the overall ASD process, we are able to add on to our prior intuition to say that non-functional requirements should be worked on together with functional requirements in most situations regardless of circumstance. Building on our initial hypothesis, we then explore different aspects of how non-functional requirements should be implemented alongside functional requirements. When looking at how much focus we should put on either functional or non-functional requirements in an agile sprint, the results of our simulations show that there is no significant difference in final performance obtained across different levels of focus on functional and

non-functional requirements (excluding the edge cases where there is no focus on either sides). However, when looking at the speed of convergence (i.e., time taken to reach the final performance), we notice that a moderate focus level without overemphasis on functional requirements gives us the final performance in the quickest amount of time. This leads us to conclude that in order to complete the project to its best potential in the quickest amount of time, a moderate focus should be placed on functional requirements. Last but not least, we explore the temporal aspect of working on non-functional requirements (i.e., introducing non-functional requirements from the beginning or later on). From our results obtained we can observe that when project complexity increases, the final performance obtained becomes higher when the non-functional requirements are introduced at a later stage. As such, we come up with the proposition that non-functional requirements should be implemented right from the beginning of the ASD project when overall complexity of the project is low, whereas non-functional requirements should be implemented later on in the project when overall complexity of the project is high.

4.2. Limitations

This paper is not without limitations. Despite the advantages of using simulation models to study complex and nuanced scenarios, it is ultimately still a stylized theoretical model that will still require validation though empirical data and testing. In addition, when using the *NK* landscape to model an ASD project, we have a fairly simplistic take on the entire ASD process (i.e., an incremental landscape search), when in reality software development is a much more complex and dynamic process (Doherty and King 2005).

With regards to the results derived from the simulations and the propositions proposed, there are also limitations to how we are able to perceive these conclusions. As mentioned in the respective sections, the different definitions and implementation techniques set out by different teams with regards to non-functional requirements would influence how we perceive the results of our simulations and the propositions that we have put forth.

In addition, with regards to the study on the introduction of non-functional requirements, a limitation of the experiment would be that we are unable to capture the notion of technical debt build up in the simulation. With the lack of focus on non-functional requirements leading to a quicker build up in technical debt (Rios et al. 2019, Jarzębowicz and Weichbroth

2021), an additional tradeoff of having to balance the quick buildup of technical debt versus holding off the focus on non-functional requirements would be present when overall project complexity increases.

Nonetheless, what we have presented are generalised results and propositions that provide us with a strong foundation and baseline for ASD teams to work with. These results also provide us with valuable insights that can act as building blocks for further theoretical and empirical research.

4.3. Recommendations for Further Work

Despite the limitations mentioned previously, our study and the results produced has allowed us to provide valuable insights that would enable further research into the topic of prioritisation of functional and non-functional requirements in agile software development.

For future research, it will be beneficial for us to set up our framework on how non-functional requirements are elicited and implemented as a benchmark for the setup of our simulation model. This would allow us to adopt a less generalised approach when studying the impacts of various prioritisation strategies and to come up with more conclusive results that are catered to different types of ASD projects and ASD teams. Modifications to the simulation model used can also be introduced to increase the robustness of the simulation and further emulate ASD projects in reality. For example, a non-binary interdependence that can capture the varying levels of dependencies between functional and non-functional requirements could be configured to better capture the unique interdependencies that exist between functional and non-functional requirements.

5. References

Aguilar, J.A., Garrigós, I., Mazón, JN., Zaldívar, A. 2012 “Dealing with Dependencies among Functional and Non-functional Requirements for Impact Analysis in Web Engineering,” Computational Science and Its Applications ICCSA 2012, Lecture Notes in Computer Science, vol. 7336.

Amorndettawin, M., & Senivongse, T. 2019. “Non-functional Requirement Patterns for Agile Software Development,” ICSEB 2019.

Anisetti. M., Ardagna. C.A., Damiani E., Gaudenzi. F., and Jeon. G. 2020 “Cost-effective Deployment of Certified Cloud Composite Services,” J. Parallel Distrib. Comput., vol. 135, pp. 203–218.

Behutiye. W.N., Rodríguez. P., Oivo. M., Tosun. A. 2017 “Analyzing the Concept of Technical Debt in the Context of Agile Software Development: A Systematic Literature Review,” in Information and Software Technology, vol. 82, pp. 139–158.

Cao. L., and Ramesh. B. 2008 “Agile Requirements Engineering Practices: An Empirical Study,” in IEEE Software, vol. 25, no. 1, pp. 60-67.

Davis, J., Bringham, C., and Eisenhardt, K. 2007. “Developing Theory Through Simulation Methods,” Academy of Management Review (32:2), pp. 480-499.

Dewi. M., Didar. Z., and Nur. N. 2009 “Managing Conflicts Among Non-functional Requirements,” University of Technology, Sydney.

Doherty, N. F., and King, M. 2005. “From Technical to Socio Technical Change: Tackling the Human and Organizational Aspects of Systems Development Projects,” European Journal of Information Systems (14:1), pp. 1-5.

Ethiraj, S. and Levinthal, D. 2009. “Hoping for A to Z While Rewarding Only A: Complex Organizations and Multiple Goals,” Organization Science. 20. 4-21.

Hahn, J. and Lee, G. 2021 "The Complex Effects of Cross-Domain Knowledge on IS Development: A Simulation-Based Theory Development," MIS Quarterly, 45(4), pp. 2022-2054.

Haki, K., Beese, J., Aier, S., and Winter, R. 2020. “The Evolution of Information Systems Architecture: An Agent-Based Simulation Model,” MIS Quarterly (44:1), pp. 155-184.

Heumesser. N., Trendowicz. A., Kerkow. D., Gross H., and Loomans. L. 2003 “Essential and Requisites for the Management of Evolution - Requirements and Incremental Validation,” Information Technology for European Advancement, ITEA-EMPRESS consortium.

IEEE Std 830-1998. 1998 “IEEE Recommended Practice for Software Requirements Specifications,” IEE Transactions on Software Engineering, vol., no., pp.1-40.

Jarzębowicz, A., and Weichbroth, P. 2021 “A Qualitative Study on Non-Functional Requirements in Agile Software Development,” IEEE Access, vol. 9, pp. 40458-40475.

Kauffman, S.A. 1993. The Origins of Order: Self-Organizing and Selection in Evolution. New York: Oxford University Press.

Kauffman, S.A., and Weinberger, E.D. 1989. “The NK Model of Rugged Fitness Landscapes and Its Application to Maturation of the Immune Response,” Journal of Theoretical Biology (141:2), pp. 211- 245.

Langlois, R.N. 2002. “Modularity in Technology and Organization,” Journal of Economic Behavior & Organization (49:1), pp. 19-37.

Levinthal, D.A. 1997. “Adaptation on Rugged Landscapes,” Management Science (43:7), pp. 934-950.

Lim, E., Taksande, N. and Seaman, C. 2012, “A Balancing Act: What Software Practitioners Have to Say about Technical Debt,” IEEE Software, vol. 29, no. 6, pp. 22-27.

Loucopoulos, P., Sun, J., Zhao, L., and Heidari, F. 2013. “A Systematic Classification and Analysis of NFRs,” 19th Americas Conference on Information Systems, pp. 1- 5.

Maiti, R.R., and Mitropoulos, F.J. 2017. “Prioritizing Non-Functional Requirements in Agile Software Engineering,” Proceedings of the SouthEast Conference.

Narduzzo, A., and Rossi, A. 2003. “Modular Design and the Development of Complex Artifacts: Lessons from Free/Open Source Software,” ROCK Working Papers 21, Department of Computer and Management Sciences, University of Trento, Italy.

Nerur, S., and Balijepally, V. 2007. “Theoretical Reflections on Agile Development Methodologies,” Communications of the ACM (50:3), pp. 79-83.

Nguyen, Q. L., 2009 “Non-functional Requirements Analysis Modeling for Software Product Lines,” 2009 ICSE Workshop on Modeling in Software Engineering, pp. 56-61

Odeh, M. and Kamm, R. 2003 “Bridging the Gap between Business Models and System Models,” Information and Software Technology, no. 45, pp. 1053-1060.

Ramasubbu, N., Kemerer, C.F. 2015 “Technical Debt and the Reliability of Enterprise Software Systems: A Competing Risks Analysis,” Management Science 62(5):1487-1510.

Rao, A., and Gopichand, M. 2011 “Four Layered Approach to Non-Functional Requirements Analysis,” CoRR. abs/1201.6141.

Rios, N., Mendonça, M.G., Seaman, C.B., and Spínola, R.O. 2019. “Causes and Effects of the Presence of Technical Debt in Agile Software Project,” AMCIS.

Rivkin, J. and Siggelkow, N. 2003. “Balancing Search and Stability: Interdependencies among Elements of Organizational Design,” Management Science. 49(3) 290–321.

Rivkin, J. W., and Siggelkow, N. 2007. “Patterned Interactions in Complex Systems: Implications for Exploration,” Management Science (53:7), pp. 1068-1085.

Tabassum. M.R., Siddik. M.S., Shoyaib. M., and Khaled. S.M. 2014 “Determining Interdependency Among Non-functional Requirements to Reduce Conflict,” 2014 International Conference on Informatics, Electronics & Vision (ICIEV), pp.1-6.

Werner, C., Li, Z.S., Lowlind, D., Elazhary, O., Ernst N., and Damian, D. 2022 “Continuously Managing NFRs: Opportunities and Challenges in Practice,” in IEEE Transactions on Software Engineering, vol. 48, no. 7, pp. 2629-2642.

World Health Organization. 2021. “High-level Functional and Non-functional Requirements,” Digital Adaptation Kit for Antenatal Care: Operational requirements for implementing WHO recommendations in digital systems, pp. 72–77.

Xia, W., and Lee, G. 2004. “Grasping the Complexity of IS Development Projects,” Communications of the ACM (47:5), pp. 68-74. (22:1), pp. 45-83.

Xia, W., and Lee, G. 2005. “Complexity of Information Systems Development Projects: Conceptualization and Measurement Development,” Journal of Management Information Systems

Yeo, Y. and Hahn, J. 2014 “The Role of Project Modularity in Information Systems Development” in Proceedings of the 2014 International Conference on Information Systems, 8.

Yusop. N., Zowghi. D., and Lowe. D. 2008 “The Impacts of Non-functional Requirements in Web System Projects,” International Journal of Value Chain Management vol. 2, pp. 18-32.

6. Appendix A – Source Code

Source code and visualisation notebooks can be accessed at
https://github.com/OoiJunHao/CP4101_NFR_FR_Prioritisation

7. Appendix B – Sample Requirements

Table B1. Sample of Functional and Non-functional Requirements for Antenatal Care Digital Tracking and Decision-support System (World Health Organisation 2021)		
Type	Category	Requirement Description
Functional	Create client record	To be able to enter identification information
Functional	Validate client details	As a health worker or clerk, I want to be able to update demographic information and retain previous history of updated information
Functional	Scheduling	As a health worker, I want to see a schedule of available days
Functional	Scheduling	As a health worker, I want to be able to input custom schedules to allow for contacts on specific days and times, account for holidays, etc.
Non-functional	Security – confidentiality	Provide encrypted communication between components
Non-functional	Security – audit trail and logs	Log system logins and logouts
Non-functional	Scalability	Be able to accommodate at least [x number of] health-care facilities
Non-functional	Scalability	Be able to accommodate at least [x number of] concurrent users
Non-functional	Interoperability	Exchange data with other approved systems
Non-functional	Interoperability	Link with insurance systems to verify eligibility and submit claims