DISSERTATION

DEVELOPMENT AND QUASI-EXPERIMENTAL STUDY OF THE SCRUM MODEL-BASED SYSTEM ARCHITECTURE PROCESS (SMBSAP) FOR AGILE MODEL-BASED SOFTWARE ENGINEERING

Submitted by

Moe Huss

Department of Systems Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2023

Doctoral Committee:

Advisor: Daniel R. Herber
Co-Advisor: John M. Borky

Erika Miller
Paul Mallette

ABSTRACT

DEVELOPMENT AND QUASI-EXPERIMENTAL STUDY OF THE SCRUM MODEL-BASED SYSTEM ARCHITECTURE PROCESS (SMBSAP) FOR AGILE MODEL-BASED SOFTWARE ENGINEERING

Model-Based Systems Engineering (MBSE) is an architecture-based software development approach. Agile, on the other hand, is a light system development approach that originated in software development. To bring together the benefits of both approaches, this research is divided into two stages. The first stage proposes an integrated Agile MBSE approach that adopts a specific instance of the Agile approach (i.e., Scrum) in combination with a specific instance of an MBSE approach (i.e., Model-Based System Architecture Process — "MBSAP") to create an Agile MBSE approach called the integrated *Scrum Model Based System Architecture Process (sMBSAP)*. The proposed approach was validated through an experimental study that developed a health technology system over one year, successfully producing the desired software product. This work focuses on determining whether the proposed sMBSAP approach can deliver the desired *Product Increments* with the support of an MBSE process. The interaction of the *Product Development Team* with the MBSE tool, the generation of the system model, and the delivery of the *Product Increments* were observed. The results showed that the proposed approach contributed to achieving the desired system development outcomes and, at the same time, generated complete system architecture artifacts that would not have been developed if Agile had been used alone. Therefore, the first contribution of this stage lies in introducing a practical and operational method for merging Agile and MBSE. In parallel, the results suggest that sMBSAP is a middle ground that is more aligned with federal and state regulations, as it addresses the technical debt concerns.

The second stage of this research compares *Reliability of Estimation*, *Productivity*, and *Defect Rate* metrics for sprints driven by Scrum versus sMBSAP through the experimental study in stage

1. The quasi-experimental study conducted ten sprints using each approach. The approaches were then evaluated based on their effectiveness in helping the *Product Development Team* estimate the backlog items they can build during a time-boxed sprint and deliver more Product Backlog Items (PBI) with fewer defects. The *Commitment Reliability* (CR) was calculated to compare the *Reliability of Estimation* with a measured average Scrum-driven value of 0.81 versus a statistically different average sMBSAP-driven value of 0.94. Similarly, the average *Sprint Velocity* (SV) for the Scrum-driven sprints was 26.8 versus 31.8 for the MBSAP-driven sprints. The average *Defect Density* (DD) for Scrum-driven sprints was 0.91, while that of sMBSAP-driven sprints was 0.63. The average *Defect Leakage* (DL) for Scrum-driven sprints was 0.20, while that of sMBSAP-driven sprints was 0.15. The t-test analysis concluded that the sMBSAP-driven sprints were associated with a statistically significant larger mean CR, SV, DD, and DL than that of the Scrum-driven sprints. The overall results demonstrate formal quantitative benefits of an Agile MBSE approach compared to Agile alone, strengthening the case for considering Agile MBSE methods within the software development community. Future work might include comparing Agile and Agile MBSE methods using alternative research designs and further software development objectives, techniques, and metrics. Future investigations may also test sMBSAP with non-software systems to validate the methodology across other disciplines.

ACKNOWLEDGEMENTS

I would like to thank a few people who, without their roles, contributions, and guidance, I wouldn't be writing these lines today. I would like first to thank Dr. Butler. When Dr. Butler agreed to be my advisor, he gave me the opportunity to pursue this journey. I am so grateful for his guidance and the time that he spent with me.

I am also thankful to Dr. Butler because he introduced me to Dr. Mallette, who was very kind to accept joining my Ph.D. committee and was very generous with his time and supporting words when I was sharing my research progress. I also want to thank Dr. Miller, who was kind enough to join my Ph.D. committee. I would also like to thank Dr. Miller again for a great engineering risk management course.

I don't have enough words to say about Dr. Borky. When I joined the program in 2018, Dr. Borky was not accepting advisees. But I was fortunate to meet Dr. Borky in Dr. Sega's office and spend some time discussing my interest in enterprise architecture. Between then and 2021, Dr. Borky was my unofficial advisor and mentor. In 2021, Dr. Borky made an exception and accepted to be officially my co-advisor. What I learned from Dr. Borky has changed my career and perception forever. I am, without a doubt, grateful that Dr. Sega introduced me to Dr. Borky.

I am also so thankful to Dr. Borky because he suggested that Dr. Herber be my advisor. Without Dr. Herber's guidance, support, and advice, I would not have reached this milestone. I learned a lot of things from Dr. Herber in the past two years, watching his extraordinary attention to detail, ability to concise data, thinking of all possible gaps, and addressing them.

I am so thankful for the role that Prof. Batchelor played in guiding me while applying for the Ph.D. in Systems Engineering program and introducing me to Dr. Butler. I would also like to thank my other professors in the Ph.D. in Systems Engineering program, Dr. Simske and Dr. Cale. Last but not least, I would like to thank Ingrid Bridge, who has been extremely helpful, responsive, and supportive throughout my journey in the program.

DEDICATION

To my wife, daughter, and son. Without your love, support, and patience, I could not have reached this milestone. Thank you.

To my late father. I know you would have been so proud of me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction[1]

The increasing complexity of software systems makes the risk of failure higher [4]. Unfortunately, software projects often do fail [5]. Model-Based Systems Engineering (MBSE) is well known for managing complexity during system development processes [6]. The fundamental concept of MBSE is to have a structured system description in a system model and link the relevant data required for creating and assembling the various artifacts during the system development process [7]. The goal is to use models across the development process lifecycle to explain relevant concepts of each system view. The generation of executable code from implementation-level models is another opportunity that has attracted much interest over the last few years [8]. MBSE for information-intensive systems could be considered an attempt to have a knowledge hub for the software development lifecycle. Additionally, the definition of a system model requires the use of a formal, standardized language, such as Unified Modeling Language (UML) [9]. Leveraging individual intellectual capabilities in software engineering is one of the many opportunities that MBSE may provide. However, it also introduces additional technical and organizational challenges, which have impacted its wide adoption [10, 11].

MBSE faces several challenges that hinder its wide adoption [11]. Several researchers discussed the challenges of implementing MBSE [12–14]. Some of these challenges are identified as related to getting support from leadership, cultural change, specific infrastructure, and tool development [12]. Accordingly, these challenges have cast doubt on implementation time and value to customers. Additionally, the time and effort required to formally describe the system using a standard modeling language are among the main constraints of widely adopting MBSE. Moreover, MBSE is known to be well-suited and achieve results for large-scale projects [15]. Accordingly, searching for alternative approaches, in numerous instances, seems necessary.

[1]Parts of this chapter previously appeared in [2, 3].

During the nineties, a number of lightweight software development methods evolved in response to the prevailing heavyweight methods that were perceived as overly regulated and planned. These software methods are collectively referred to as Agile software development methods. One of the benefits of Agile is its alignment with systems engineering models such as the ISO/IEC/IEEE 15288 standards [16] and the widely-used V-model [17]. As software systems increase in size and complexity, new approaches to abstraction, concurrency, and uncertainty must be devised [18]. Agile methodologies do provide realistic and appealing strategies to evolve systems and software engineering to address these concerns [18]. Although there has been significant adoption in recent years, discussion about software development projects not meeting expectations has been increasing [19–21]. Given that there are still challenges that hinder MBSE and Agile from unlocking their full potential, Agile MBSE presented itself as a possible resolution [18, 22–24].

Some researchers have proposed combining MBSE with Agile to leverage the benefits of both worlds in addressing the complexity of information technology systems, especially if adopted during the system architecting process [2, 22–26]. However, sources that provide considerable and persuasive data about the convergence of MBSE and Agile are still in their infancy, despite researchers' efforts. The lack of research that quantitatively compares Agile MBSE with Agile impedes the consideration of Agile MBSE for technology-based systems. Researchers have also seen the need for techniques and methods that support documentation in Agile environments [27]. A technique that makes documentation more easily writable, manageable, and updatable is needed [28, 29]. Therefore, the first stage of this study adopts a specific instance of the Agile approach (i.e., Scrum [30]) in combination with a specific instance of an MBSE approach (i.e., Model-Based System Architecture Process (MBSAP) [1]) to create an Agile MBSE approach called the integrated *Scrum Model Based System Architecture Process (sMBSAP)*.

The second stage of this research aims to compare sMBSAP [2] with Scrum. The overarching objective is to investigate how the software development performance using Agile MBSE is compared to that of Agile methodology alone when developing software systems. This stage specifically compares the *Reliability of Estimation*, *Productivity*, and *Defect Rate* of sMBSAP vs. that

of Scrum. This will be done by analyzing the results from a quasi-experimental study conducted to investigate the effects of sMBSAP and Scrum on the software development performance while developing software for a health technology system. This research contributes to the literature by explaining a practical approach to implementing Agile MBSE. The research also applies the proposed approach to a health technology system. The proposed sMBSAP attempts to narrow the gap between Agile practitioners and document-based/waterfall practitioners by simplifying and streamlining system documentation and making it easier to develop, manage, access, and share. The second contribution of this research is that it provides a quantitative comparison between Agile MBSE and Agile in an attempt to open the door for further quantitative comparisons and for consideration of Agile MBSE for technology-based systems.

The remainder of the chapter is organized into three sections. The first section (Section 1.1) provides a background about the challenges that face software systems (Section 1.1.1). It also provides an overview of Software Development Lifecycle (Section 1.1.2), which will be discussed further in Chapter 2. Finally, it provides a background of the three software development performance objectives (Section 1.1.3), specifically, *Reliability of Estimation*, *Productivity*, and *Defect Rate*, which will be discussed in conducting the experiment as explained in the research methodology in Chapter 3. The second section (Section 1.2) explains the research purpose and the associated research question and hypotheses. The third section (Section 1.3) is dedicated to giving an overview of the health problem (Section 1.3.1), the need for a health tech system (Section 1.3.2), and finally, a description of the health tech system (Section 1.3.3).

1.1 Background

1.1.1 Challenges of Software Systems

Software systems play an essential role in society today. It penetrates people's daily lives in an unprecedented and sometimes unintended way. It determines how individuals communicate, work, learn, and perform their other daily activities. In the same way, Information Technology systems become a critical competitive element for organizations [31].

Systems and Software engineers face challenges using current systems and software engineering approaches to manage the development of these complex software systems because they are getting larger and more complex, and the risk of failure is becoming higher [4]. Unfortunately, things often do fail [5]. Several works of literature discuss the history, factors, and impact of Information Technology (IT) projects' failure [19–21]. In collaboration with the University of Oxford, McKinsey studied 5400 large IT projects exceeding $15 million. McKinsey found that, on average, large IT projects run 45% over budget and 7% behind schedule while delivering 56% less value than initially planned [32]. Flyvbjerg and Budzier found that Software projects carry the highest risk of cost and schedule overruns [33]. McKinsey summarized the reasons for large IT project failures into four categories: missing focus, content issues, skills issues, and execution issues. Examples of multi-million challenged information systems include UK National Programme for IT (£10.1bn), US Air Force Expeditionary Combat Support System ($1bn), FiReControl (£469m), MyCalPays ($373m), and others [34].

The impact of such high risk may go beyond the project and threaten the organization's existence. In 2000, Kmart embarked on a $1.4 billion IT modernization project to regain its market share from Walmart and Target. By 2001, Kmart realized that the maintenance of the new system would be too expensive because of the level of system customization [35]. Accordingly, it launched a $600 million project to update its supply chain management software. In 2002, both projects went off track and contributed to Kmart's bankruptcy [35]. Other examples of multi-million challenged information systems include UK National Programme for IT (£10.1bn), US Air Force Expeditionary Combat Support System ($1bn), FiReControl (£469m), MyCalPays ($373m), and others [36]. In the past decade, many of California's most prominent tech projects have suffered massive delays and substantial cost overruns [37]. The state paid out nearly $900 million on three projects before canceling them: (1) 21st Century Project overhaul of the state's payroll system, (2) Department of Motor Vehicles upgrade of the driver's license and vehicle registration system, and (3) Judicial Council courts management system [37]. If governments and organiza-

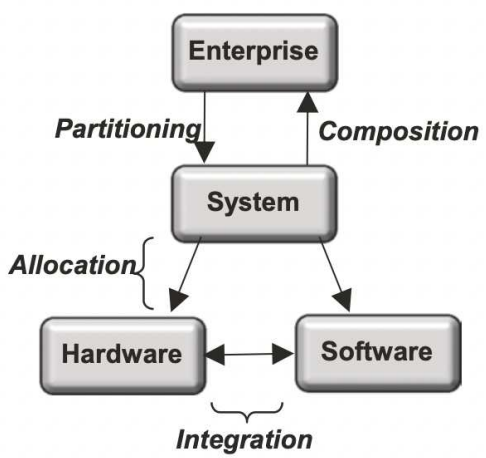


Figure 1.1: Relationships among enterprise, system, hardware, and software architectures [1].

tions are keen on optimizing their resources, new software engineering approaches will be critical to controlling costs, maintaining schedules, and ensuring value delivery.

1.1.2 Software Development Lifecycle (SDLC)

The Software Development Lifecycle (SDLC) is a process for building and maintaining software systems. Several researchers have explained that system development lifecycle models have been strongly influenced by software, and so the two terms "system" and "software" might be used interchangeably in terms of SDLC [38], especially since system development encompasses software system development [38–40]. Others have clarified that systems engineering, software systems engineering, and software engineering have different areas of focus [41].

The relationship between enterprise, system, hardware, and software engineering is illustrated in Figure 1.1 [1]. An enterprise can be broken down into the systems that comprise it. A system is constructed by assigning functionality to the hardware and software, which are then integrated to produce a solution to the business problem.

Currently, two major SDLC categories are used when managing software systems projects: (1) traditional and (2) Agile. Several commonly used traditional systems development methods are shown in Figure 1.2. Traditional systems development methods have been historically associated with Document-Based Systems Engineering (DBSE) practices [42].

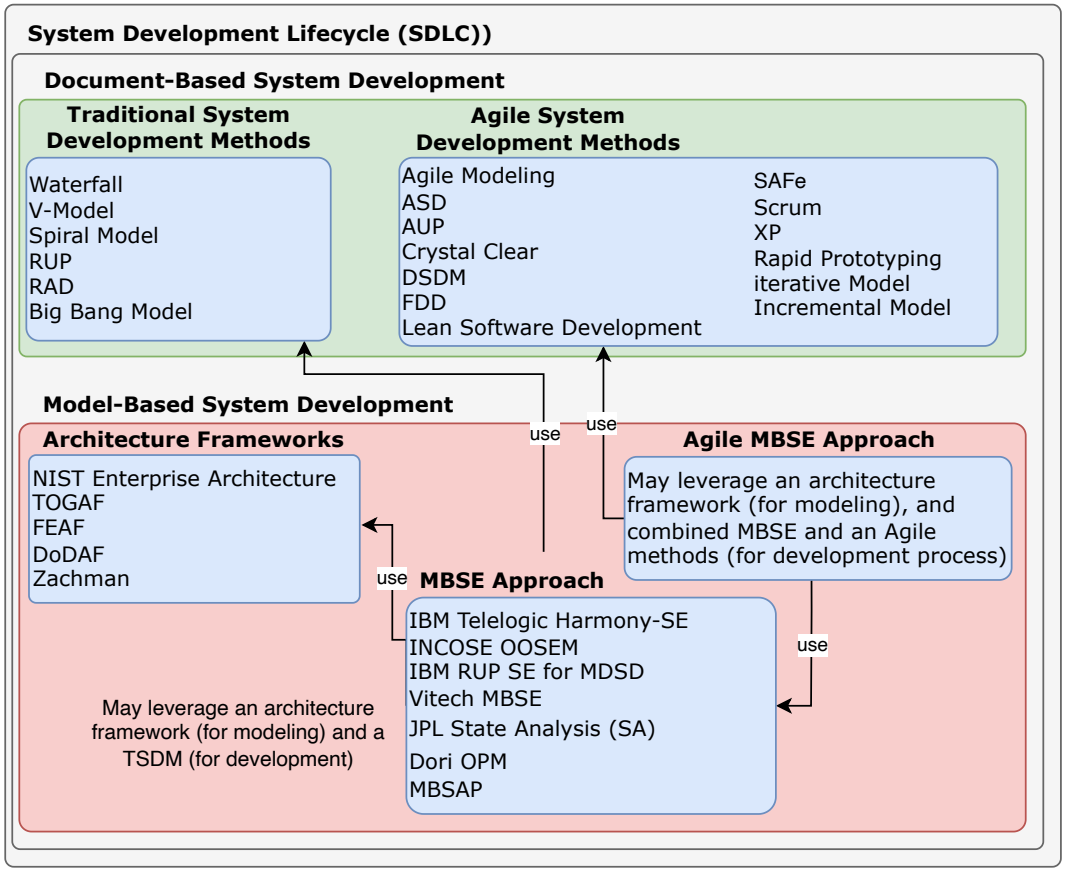


Figure 1.2: Overview of software development lifecycle models and their relationship with MBSE.

The traditional methodology is a commonly used approach by organizations whereby software development activities are completed sequentially. However, the drawbacks of traditional methodology started to impact projects' success in these organizations. For instance, the sequential approach does not allow the development team to catch defects before the testing phase, which would cause a considerable amount of rework. Additionally, it is common in software development that stakeholders do not know all the *Requirements* at the beginning of the project. Therefore, *Requirements* defined at an early phase are commonly subject to continuous change; accordingly, the project must deal with many change requests. As a result of implementing multiple change requests, from different stakeholder groups, into the system, it suffers various compatibility and integrity issues.

Some of the characteristics that make traditional systems development methodologies widely used to date in various industries include their structured nature [18], stability and predictabil-

ity [25], stringent safety, and reliability [22]. Traditional systems development methodologies, including the waterfall method, V-model, Rational Unified Process, spiral models, and others [43] require extensive planning, highly structured process, heavy documentation, and extensive up-front design [44]. Due to these characteristics, traditional systems development methodologies are often called heavy methods [45].

Implementing project management standards can improve efficiency in system development [46]. However, traditional system development methodologies are plagued with several gaps, including the inability to respond quickly to changing *Requirements* [47] and a propensity to be delivered with fewer features and functions than specified [44, 48]. Moreover, traditional system development projects have notoriously been known to be over budget and behind schedule [49–51]. Researchers report that software development projects that use traditional methods have a 15% successful completion rate and a 30% failure rate [52].

On the other hand, Agile systems development methods are the other category of SDLC that came to life to address some of the challenges associated with traditional systems development methods. The first generation of Agile approaches, sometimes called lightweight development methods [53], are also shown in Figure 1.2. Agile was first introduced in 2001 in the Agile Manifesto [54]; it introduced four core values and 12 principles as a practice-based mindset and philosophy to manage software development projects. Agile software development is an iterative and incremental approach to software development. In this approach, the *Requirements* can be changed in response to customer needs. It supports iterative development, adaptive planning, and time boxing [55]. The Agile SDLC includes *Requirements* gathering, analysis, design, coding, testing, delivery of partially implemented software, and seeking customer feedback for consideration in future iterations. Sharma et al. [55] identified ten characteristics of Agile projects: iterative, modularity, time boxing, parsimony, incremental, adaptive, convergent, collaborative, and people-oriented.

The Scrum method [56], an instance of Agile, enhances the iterative and incremental approach to delivering complex software and products. MBSE offers techniques to aid in developing com-

plex systems, aiming to improve system quality, increase velocity/Agility, improve User Experience (UX), and improve knowledge management over traditional Systems Engineering techniques. MBSAP [1], a specific implementation of the principles of MBSE, is intrinsically aligned with Agile principles. MBSAP provides a rigorous and traceable mapping of customer *Requirements*, rapid prototyping, human-in-the-loop, and technology demonstrations.

Although Agile methodologies have merits over traditional development methods, several limitations have been faced when expanding their adoption [43]. One of these limitations is that Agile methods considerably decrease the amount of required documentation [57], which is a source of subsequent issues. The lack of documentation does not align well with federal and state acquisition regulations. A report by the U.S. Government Accountability Office (GAO) identified 14 challenges related to implementing the Agile approach in federal agencies. These challenges include that government contracts may be designed with heavily structured tasks and performance checks that are not necessarily aligned with Agile methods or cadences [58]. A report [58] also highlighted the need to track several different Agile metrics, such as *Requirements* and architectures. Without tracking such metrics, the government may not have the right information for effective contract oversight [58]. The abilities of MBSE methodologies to centrally manage documentation and track *Requirements* and architectures make them a reasonable solution for aiding in complying with such regulations.

1.1.3 System Development Performance

Methodologies for developing software systems are frequently evaluated in terms of development cost, development time, and impact on quality (measured in the number of defects) [59], accuracy in estimation, and increased *Productivity* [9]. Therefore, this section sheds light on three software development performance objectives that are commonly used in software development projects.

1.1.3.1 Reliability of Estimation

Effort estimation is a vital part of managing software projects. There is plenty of research in effort estimation models, techniques, and tools [60–63]. What makes estimation in Agile Software projects unique is that software is developed incrementally, and customer feedback significantly influences the subsequent iterations. This also implies that product estimations and planning need to be done progressively. This is achieved in Scrum by planning iteratively at three levels (i.e., release planning, *Sprint Planning*, and daily planning) [64]. Common estimation techniques in Agile software development include Expert Opinion, Analogy, and Disaggregation [64]. Planning Poker [65] is another widely used technique, especially in Scrum [66, 67]. A study about using Planning Poker for user story estimation found that Planning Poker produced more accurate estimates than unstructured group estimates [68]. Other studies found that Planning Poker in Scrum projects yielded less optimistic and more accurate estimates than a statistical combination of individual expert estimates [69, 70].

Estimation accuracy in software development has also been investigated. An in-depth study on 18 different companies and project managers of 52 different projects found that average effort overruns are 41% and that the estimation performance has not changed much in the last 10–20 years [71].

Grimstad et al. proposed a framework for improving the analyses of software cost estimation error [72], arguing that a lack of clarity and precision in the use of estimation terms reduces the interpretability of estimation accuracy results [72]. Another study found that in projects where experience data was utilized in the estimation process, a lesser magnitude of effort overruns was observed [73]. The study also suggested that the use of a checklist increased estimation accuracy [73]. However, the utilization of an estimation model did not appear to have a substantial impact [73].

Several researchers studied and analyzed the use of the *Commitment Reliability* CR metric (referred to as say/do ratio, estimation accuracy, or simply planned vs. actuals) in software projects [74–77]. Although Agile teams use other *Reliability of Estimation* metrics, CR is the most widely

used metric in practice. The CR metric used by Agile teams is identical to the Earned Value (EV) metric, part of the Earned Value Management (EVM) methodology used in traditional project management.

The EV index is the ratio between the Planned Value (PV) and the Actual Cost (AC) [78]. EVM is a methodology for measuring project performance and progress [79], started by the U.S. Department of Defense (DoD) in the 1960s, and is now required for many U.S. government projects and programs [80–83]. The need for government organizations to report on their internal control over financial resources contributed to the increased interest in applying and implementing EVM [84, 85]. The EVM quickly found its way to other industries such as construction [86–88] and manufacturing [89, 90].

Carson et al. introduced the concept of Performance-Based EV and applied it to developing a systems engineering metrics architecture to measure systems engineering effectiveness [91]. The idea of leveraging the proven EV methodology in software development is not new. In 1992, Kadary et al. described a method to implement the EV to measure software development [92]. Several works of recent literature discuss the implementation of EVM in Agile software development, in general, [93–96] and in Scrum projects, specifically [97].

1.1.3.2 Productivity

One of the reasons Scrum is widely used in Agile software development is the positive correlation between Scrum and project *Productivity* [98]. Using high-frequency tracking tools such as burnup charts to monitor issues during sprints allows the *Product Development Team* to make course corrections if needed before problems exacerbate. Monitoring *Productivity* provides the *Development Team* capabilities to (1) estimate how much scope can be delivered in future iterations, (2) estimate a date for a given amount of scope to be delivered, and (3) understand the *Productivity* capacity of the *Team* when committing to future work. These capabilities will have a positive impact on completing the project within time and budget while delivering the promised value. Other outcomes such as product quality, client satisfaction, cost reduction, and team motivation have been found to be associated with *Productivity* [98]. Wagner et al. [99] conducted

a systematic review of *Productivity* factors in software development. They divided the *Productivity* factors into technical and soft factors, with the identified 24 technical factors broken down into three categories: product, process, and development environment. They also identified 27 soft factors divided into five categories: corporate culture, team culture, capabilities and experience, environment, and project. In a systematic review of *Productivity* factors in software development, it was found that soft factors are often not analyzed with equal detail as more technical factors [99]. It has been argued that the factors influencing *Productivity* depend on the business domain the software is produced for [100]. It has also been found that more than a third of the time a typical software developer spends is not concerned with technical work [101] and that reusability has a significant effect on *Productivity* [102].

In software development, the Count of Lines of Code ($CLOC$) and Story Points (SP) (sometimes referred to as function points) are traditionally used as measures of *Productivity* [99]. *Sprint Velocity (SV)* is the number of SP completed by the *Product Development Team* in one Sprint [64]. Several researchers have studied and analyzed the use of SV in software projects [103–105]. Although Agile teams use other metrics, the most influential metrics in many of the studies are SV and effort estimate [106]. SV is a function of direction and speed [107]. Although the direction is more easily overlooked, both factors should be equally considered [107]. The challenge for teams to decide how time should be split between ceremonies and product build tasks has been investigated by researchers [108]. *Velocity Fluctuation (VF)* is a vital graph that product managers watch during *Sprint Planning* and execution because it helps them plan more accurately for future sprints. VF has been studied to identify the sources of issues arising in Scrum implementations, and it was found that fluctuations in velocity are caused by the team missing their commitments that, in turn, depend on the unaccounted complexities of different subsystems [109].

1.1.3.3 Defect Rate

IEEE defines error, defect, and failure in its standard glossary of software engineering terminology [110]. A software defect is an error in coding that causes failure in software [111]. It is a form of deficiency that causes the software to produce an incorrect, unintended, or unexpected

result [112]. An error is an incorrect action on the part of a programmer [110]. With software's growing size and complexity, software testing plays a critical role in capturing software defects. Detecting and fixing software defects are the most expensive part of the software development lifecycle [113]. Researchers investigated the increasing cost of detecting defects throughout the software development lifecycle [44]. During the development step, detecting and fixing defects cost $977 per defect; the cost increases to $7,136 (or 630%) per defect in the testing phase; then it is further doubled in the maintenance phase, reaching $14,102 [114]. It is no wonder that software defect prediction is a popular research topic in the software development field with such high penalties [115].

Defect Density (DD) is another widely used *Defect Rate* metric [116–118], which is defined as the number of defects divided by size [118]. Some researchers studied and calculated DD using *Thousands (Kilo) of Lines of Code* ($KLOC$) as a measure of size [118]. Other researchers use other size metrics such as SP [119], *User Stories* or *Product Backlog Item (PBI)* [64, 120], and developer's effort (measured by person-hours or man-days) [121]. The *Defect Leakage* (DL) [122–125], another known *Defect Rate* metric, is the ratio of defects found during testing vs. defects found during the next phase.

1.2 Research Purpose

Although few researchers discuss the integration between MBSE and Agile and their qualitative benefits, the literature that provides practical method for its implementation and quantitative comparative analysis is still limited. This scarcity hinders considering Agile MBSE as an option for technology-based systems. The literature sparsity on this problem leads to the following research question:

Comparing sMBSAP and Scrum, are there measurable benefits to software development performance when using one approach over the other?

Return-on-investment (ROI) for software system development methodologies, including Agile and MBSE, is commonly measured in terms of development cost, development time, and impact on

quality (measured in the number of defects) [59], estimation accuracy, and increased *Productivity* [9]. The purpose of this research has two folds. First, introduce a practical and operational method for merging Agile and MBSE. Second, conduct an experiment to compare sMBSAP [2] and Scrum, in terms of *Reliability of Estimation*, *Productivity* and *Defect Rate*. The development of a health tech system that provides dietary recommendations to users based on their health profile [2] was used as a context for this experiment. Section 1.3 provides an overview of the health tech system. In the experiment, the dependent variables were the software development performance, namely (1) *Reliability of Estimation*; (2) *Productivity*; and (3) *Defect Rate*. The categorical independent variable is the system development approach has two groups, one treatment (*sMBSAP*) and one control (*Scrum*).

Accordingly, the overarching research question is translated into three hypotheses. Each hypothesis is concerned with one software development performance objective (dependent variable). The first hypothesis corresponds to *Reliability of Estimation*; the second hypothesis corresponds to *Productivity*; and the third hypothesis corresponds to *Defect Rate*. The hypotheses are as follows:

First Hypothesis

Hypothesis H_0: sMBSAP has equivalent or better *Reliability of Estimation* relative to Scrum, as assessed by *Commitment Reliability* (CR), that is, the ratio of work delivered vs. work committed during a sprint, measured in *Story Points* (SP).

Hypothesis H_1: sMBSAP has inferior *Reliability of Estimation* relative to Scrum, as assessed by *Commitment Reliability* (CR), that is, the ratio of work delivered vs. work committed during a sprint, measured in *Story Points* (SP).

Second Hypothesis

Hypothesis H_0: sMBSAP has equivalent or better *Productivity* relative to Scrum, as assessed by (a) *Sprint Velocity* (SV), that is, the amount of work delivered during a sprint, measured in Story

Points (SP), (b) *Velocity Fluctuation* (VF) that is the deviation from the average *Sprint Velocity*, and (c) *Count of Lines of Code* ($CLOC$) $per\ hour$.

Hypothesis H_1: sMBSAP has inferior *Productivity* relative to Scrum, as assessed by (a) *Sprint Velocity* (SV), that is, the amount of work delivered during a sprint, measured in Story Points (SP), (b) *Velocity Fluctuation* (VF) that is the deviation from the average *Sprint Velocity*, and (c) *Count of Lines of Code* ($CLOC$) $per\ hour$.

Third Hypothesis

Hypothesis H_0: sMBSAP has equivalent or lower *Defect Rate* relative to Scrum, as assessed by (a) *Defect Density* (DD), that is, the number of defects found within *Sprint Backlog Items (SBIs)* or *Thousands (Kilo) of Lines of Code* ($KLOC$), and (b) *Defect Leakage* (DL), that is, the ratio of defects found during testing vs. defects found during the next phase, measured in defects count.

Hypothesis H_1: sMBSAP has higher *Defect Rate* relative to Scrum, as assessed by (a) *Defect Density* (DD), that is, the number of defects found within *Sprint Backlog Items (SBIs)* or *Thousands (Kilo) of Lines of Code* ($KLOC$), and (b) *Defect Leakage* (DL), that is, the ratio of defects found during testing vs. defects found during the next phase, measured in defects count.

1.3 Overview of Health Tech System

1.3.1 The Significant Impact of Health Problems on Society

COVID-19 has put health on the agenda of every nation. The direct connection between the global population's health and our economic prosperity has been so visibly demonstrated. Research by the McKinsey Global Institute [126] shows that making prudent investments in the health of the world's population can dramatically improve people's quality of life, protect against downside risks such as pandemics, and lead to significant economic returns from increased output and productivity. The upside for the economy and society would be enormous: a $12 trillion economic opportunity, hundreds of millions of lives saved, and better health across the global population.

McKinsey Global Institute estimate that poor health reduces global Gross Domestic Product (GDP) by 15% each year — about twice the pandemic's likely negative impact in 2020 — from premature deaths and lost productive potential among the working-age population. As a society, we are facing significant health problems.

- The United States ranks 45 in life expectancy among the world countries [127].
- We have a workforce plagued with absenteeism and reduced productivity because of chronic health problems, including depression. The The Center for Disease Control and Prevention (CDC) reports that absenteeism in the US costs employers $225.8 billion annually in productivity losses [128].
- According to the The Center for Disease Control and Prevention (CDC), 90% of the nation's 3.5 trillion in annual health care expenditures are for people with chronic and mental health conditions [129]. Preventing chronic diseases or managing symptoms when prevention is not possible can reduce these costs.

1.3.2 The Need for the Health Technology System Selected for the Experiment

With the accelerated technological development, many companies and developers have been developing digital health applications and systems. These systems help users track their calories and activity level and provide high-level recommendations to decrease their sugar or fat intake based on their weight and activity level. It also encourages users to have a balanced diet by tracking their fiber and macronutrients (carbohydrates, fat, and protein) [130] to reach an Acceptable Macronutrient Distribution Range (AMDR) [131].

Many researchers now confirm that diet can play a significant role in controlling many diseases [132]. Health is becoming increasingly important for Americans, with 85% considering themselves healthy eaters and 20% following a specific diet or lifestyle [133]. However, there is no known solution to address the need for personalized nutrition tailored to one's medical profile for several reasons:

1. Most Users do not have the knowledge to interpret their medical reports. DNA tests, macrobiotic and metabolomic profiles, lab results, and medical reports do not provide specific advice on what diet to use to mitigate the risk of potential health conditions. Hence, adults tend to turn to online resources to fill their knowledge gaps and obtain actionable advice.

2. Most users struggle to comprehend food labels. Grocery stores carry up to 50,000 grocery items, each with its different ingredients, which puts much pressure on the person who decides to take control of their dietary habits. A study conducted by OnePoll in conjunction with Crispy Green looked into the habits and thoughts of 2,000 Americans and found that 77% are struggling to understand and trust food labels [133].

3. Most users do not have a way to monitor the impact of what they eat on their health to make corrective actions along the way.

Despite the relevance of the existing applications and systems, no system has been found to provide recommended diets for users based on their health profiles, specifically for those with one or more (comorbid) medical conditions. Given this gap and the potential value of a practical solution, developing a prototype of this system is selected as a case study.

A study examined United States adults' understanding of a Nutrition Facts panel (NFP) [134], which requires health literacy, and the association between label understanding and dietary behavior. It was found in that study that approximately 24% of people could not determine the calorie content of the entire ice cream container, 21% could not estimate the number of servings equal to 60 g of carbohydrates, 42% could not estimate the effect on the daily calorie intake of one serving, and 41% could not calculate the daily percentage value of calories in a single serving.

The American Heart Association (AHA) conducted another study to understand consumer attitudes towards healthy labels and different food packaging aspects that drive purchase behavior. Although customers usually read food packaging labels, the study found that consumers sometimes struggle to identify healthy food options. Only 28% of 1,017 total US consumer respondents reported that it is easy to find healthy food [135]. Another study [136] concluded that the pres-

ence of Nutrient Content Claims (NCCs) on a fortified snack food product increased the perceived healthfulness of that product, perceptions of the presence of healthful nutrients, and intentions to consume the product. Several other studies discuss how consumers maybe lead to making poor dietary decisions, such as those conducted by Verrill [137, 138] and Labiner-Wolfe [139].

A study conducted on patient perceptions of receiving test results via online portals found that online portals are not currently designed to present test results to patients in a meaningful way. Specifically, it was found that nearly two-thirds (63%) did not receive any explanatory information or test result interpretation when they received the result, and 46% conducted online searches for further information about their result [140]. The risk in such instances is that people may seek medical advice from unreliable sources such as Yahoo! Q&A [141].

Several researchers have discussed the use of technology in health and diet [132, 142–145]. However, this domain is still growing and requires more research [146, 147]. The literature on using technology to provide diet recommendations is minimal. Little [148] is one of the very few researchers who discussed the use of technology to manage diet-related chronic diseases. The literature on using technology to assess user health to provide diet recommendations is very scant.

An online survey, followed by semi-structured interviews, was conducted among 229 potential users of the system aged 18 and older found:

- 60% of the survey participants reported that their medical records are either "not at all helpful" or "slightly helpful" in guiding the user to what specific grocery items fit their medical condition/health objective they should buy.

- 67% of the survey participants reported either "comfortable" or "very comfortable" sharing their medical reports with an application that guides them to the recommended grocery items that fit their specific medical condition/health objectives.

- 84% of the survey participants reported that they either "probably will" or "definitely will" use a mobile application that helps them shop for grocery items tailored to their medical condition/health objectives if such an application exists.

Figure 1.3: Problems people face with diet and system proposed solution.

Based on the scarcity of literature and the promising survey results, the system selected for this experimental study is a system that provides diet recommendations based on the user's health profile.

1.3.3 System Description and Scope

The system aims at solving three problems that people face with diet (1) analyzing medical records, (2) interpreting food labels, and (3) monitoring the impact of diet on their health. The system attempts to solve these problems through the steps shown in Figure 1.3.

The system prototype is a web-based application. The user shall be able to register and create login credentials. The user, then, shall complete a health risk assessment. This assessment will allow the system to develop a health profile for the user. At the end of the health risk assessment, the user shall view a health report that includes personalized diet and lifestyle recommendations. Screenshots of the user's value stream can be found in Chapter 2. Additional information about the health technology system can be found in Appendix E.

Although the focus of the case study is developing a system prototype, the architecture of the prototype can still be classified along the four architecture taxonomic dimensions defined by [1] and shown in Figure 1.4:

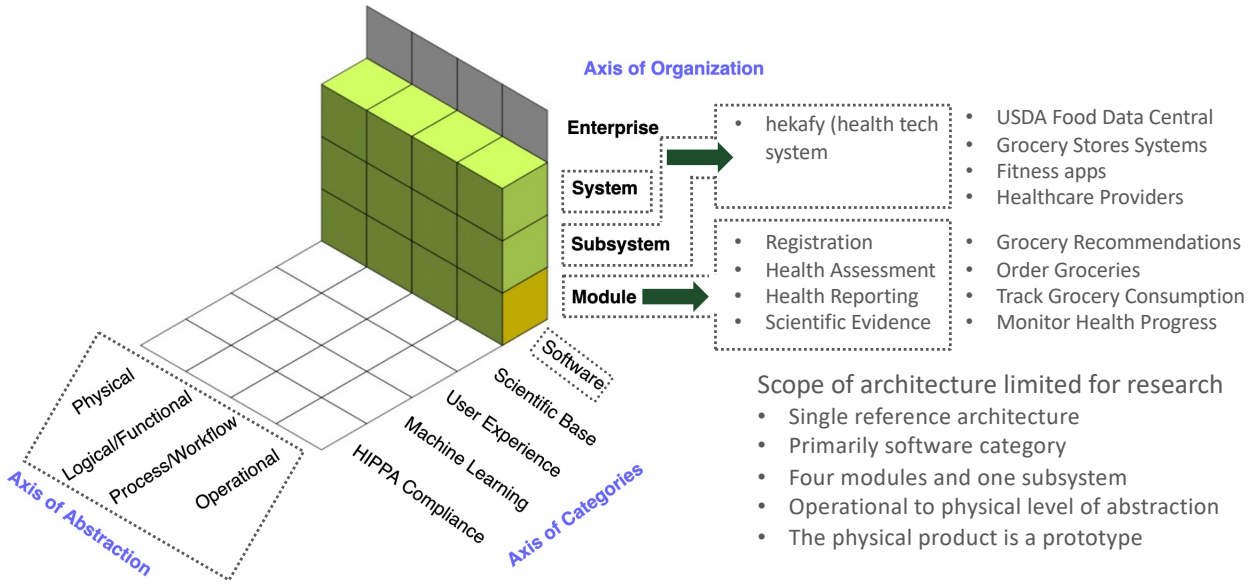


Figure 1.4: Scope of the software development for the health tech system mapped to the architecture taxonomy. [1].

- **Abstraction**: Progression of the architecture from abstract to concrete.
- **Organization**: Level of the architecture, from that of an entire enterprise to its individual subsystems.
- **Categorization**: Architectural categories to meet the requirements of various users, each with their own business processes and required levels of performance.
- **Time**: Period of time in which an architecture version is defined.

The architecture space for the first three architecture taxonomic dimensions is shown in Figure 1.4. The yellow volume shown represents the focus of the case study. The axis of organization spans from the top-level company, where the system is developed, down to the individual system modules. For the research project's purpose, the enterprise level is outside this project's scope. The axis of categories defines the key *Functional Domains* relevant to the project and key to its architecture. The system architecture categories include the following *Functional Domains*:

- **Software**: Front-end (FE) and Back-end (BE) of the web application

- **Scientific base:** literature, randomized clinical trials, nutritional practice guidelines, clinical practice guidelines
- **User Experience (UX)**: interaction, wireframes, user research, scenarios, graphics, layouts of the mobile application
- **Machine Learning (ML)**: data, decision model, decision engine, decision modeling
- **Health Insurance Portability and Accountability Act (HIPPA) Compliance**: medical records security, user privacy, authentication, data encryption

As shown on the axis of categories in Figure 1.4, the software domain is the research project's focus. The axis of abstraction spans from operational to physical definition. The system time dimension could be defined by the six phases of the user's feedback loop [149]. Architecting, an initial, feasible system concept, occurs in the first iteration and specifically during the "Build" phase. The user feedback started with an idea validation survey as suggested by Blank [150]. The project continued to evaluate a range of ideas and alternative options in subsequent iterations to determine the most optimum route for the desired system [149]. The successive versions of the architecture and the resulting increments are labeled using the terms alpha version, beta version, and launch version.

The scope of the architecture was limited for this research to fit within the available project resource constraints (i.e., project team availability and schedule). The primary focus of the architectural model is to inform the sprints to deliver the planned releasable *Product Increment*, not to describe the system entirely and comprehensively. This focus will allow performing the comparative analysis between the proposed sMBSAP and the traditional Scrum.

The scope of the system case study is shown in Figure 1.4. Although the system is planned to include eight modules, it is important to note that four modules are within the scope of the research project. The four modules are:

- Registration and Authentication

- Health Assessment (HA)
- Health Reporting (HR)
- Scientific Evidence Platform (SEP)

1.4 Dissertation Organization

This dissertation proposes an integrated Agile MBSE approach called the integrated Scrum Model Based System Architecture Process (sMBSAP). Then, compares software development performance for sprints driven by sMBSAP versus that of Scrum when developing a software system. The remaining chapters are organized as follows:

- Chapter 2 starts with the background of Scram (Section 2.1), MBSE (Section 2.2), MBSAP (Section 2.3), and Agile MBSE (Section 2.4), and finally presents a practical and operational method for merging Agile and MBSE into an integrated sMBSAP(Section 2.5).
- Chapter 3 discusses the experimental research methodology, including research philosophy and approach (Section 3.1) research design (Section 3.2), techniques and procedures (Section 3.3), and finally, the quality of research design (Section 3.4), which includes reliability and validity (3.4.1), mitigating threats to reliability (3.4.2), reliability, mitigating threats to validity (3.4.3), and finally mitigating threats to the statistical conclusion (3.4.4).
- Chapter 4 discusses the observations and findings related to the proposed sMBSAP (Section 4.1) and presents the results of the data analysis for the experiment (Section 4.2). The chapter concludes which additional observations (4.2.4) were captured during the experiment.
- Chapter 5 concludes the results obtained from the development of the methodology (Section 5.1), synthesis of the experiment results (Section 5.2), and provides conclusions of the research (Section 5.3). This chapter also sheds light on the research contribution (Section 5.4). Finally, it provides insight into the path forward in future work areas (Section 5.5)

to better investigate the differences between Agile and Agile MBSE from the software development performance standpoint.

Chapter 2

The Integrated Scrum Model-Based Systems Architecture Process (sMBSAP)[2]

This chapter introduces the proposed Scrum Model Based System Architecture Process (sMBSAP) approach. The first section (Section 2.1) provides a background about the Scrum methodology. The second section (Section 2.2) briefly explains Model-Based Systems Engineering. The third section (Section 2.3) is dedicated to provide a high-level overview of the Model-Based System Architecture Process (MBSAP). The fourth section (Section 2.4) sheds light on the notion of integrating Agile and MBSE. The last section introduces the proposed approach for merging Agile and Model-Based Systems Engineering (MBSE) into an integrated sMBSAP (Section 2.5).

2.1 Scrum Method

The Scrum software development process is an Agile method for managing and directing the development of complex software and products by using incremental and iterative techniques [151]. The State of Agile Report [152] revealed higher adoption of Scrum-based development in the present-day software industry compared to other methodologies. It was found that 87% of the 2022 survey participants leveraged Scrum [152].

Scrum is an enhancement of the iterative and incremental approach to delivering object-oriented software, a concept that was initially introduced by Pittman [153] and further extended by Booch [154]. It may use the same project team structure explained by Graham [155], but it drives the team process in a new way [56]. The popular sport of rugby, in which two teams compete against each other, is where the term "Scrum" first appeared. Takeuchi and Nonaka [156] were the first to use rugby strategies to describe hyper-productive development processes in Japan. Three team-related strategies from rugby are adopted into the Scrum approach, including a holistic team approach,

[2]Parts of this chapter previously appeared in [2, 3].

constant interaction among team members, and unchanging core team members. Schwaber and Sutherland, the fathers of the Scrum approach [157], first introduced it to the public at the OOPSLA conference in 1996 [56]. The Scrum method uses an empirical process control supported by transparency (visibility), inspection, and adaptation [30].

The Scrum method includes three main components: roles, ceremonies, and artifacts [30]. The Scrum processes are grouped into five phases: initiate, plan and estimate, implement, review and retrospect, and release [158]. In addition, there are three distinct roles in the Scrum process: the *Product Owner*, the *Scrum Team*, and the *Scrum Master* [159]. The Scrum method includes periodic meetings known as ceremonies, which include *Sprint Planning*, *Daily Scrum (Standup)*, *Sprint Review*, and *Sprint Retrospective* [24, 95, 160, 161]. In addition to the Scrum roles and ceremonies, the Scrum process delivers three primary artifacts, namely, the *Product Backlog*, the *Sprint Backlog*, and the *Product Increment* [24, 95, 160, 161]. A high-level overview of the Scrum ceremonies, artifacts, and phases is shown in Figure 2.1.

Scrum artifacts are the key information a *Scrum Team* uses and generates during the sprints to show their progress to stakeholders. Some researchers even claimed that the code itself should act as a document [57]. Agile practitioners devalue comprehensive documentation and *traceability*; therefore, architecture artifacts, such as *Use Case Diagrams*, *Data Models*, and *Requirement Traceability*, are not typical in Scrum methodology [25, 159]. *User Stories* are used to capture scenarios or system *Behavior* by describing how a user interacts with the system [25]. With the Scrum approach, information elements are captured in an Agile tool (such as Jira [162], and ClickUp [163]) and amongst separate, independent documents by using Microsoft Office tools (rather than a primary, integrated system model as a single source of truth, as is the case in MBSE approaches).

Below are definitions of key Scrum terms and artifacts:

- **Sprint**: A short, time-boxed period when a *Scrum Team* works to complete a set amount of work [24]. Sprints are at the heart of Scrum and methodologies, and getting sprints right, helps teams ship better software with fewer issues [164, 165].

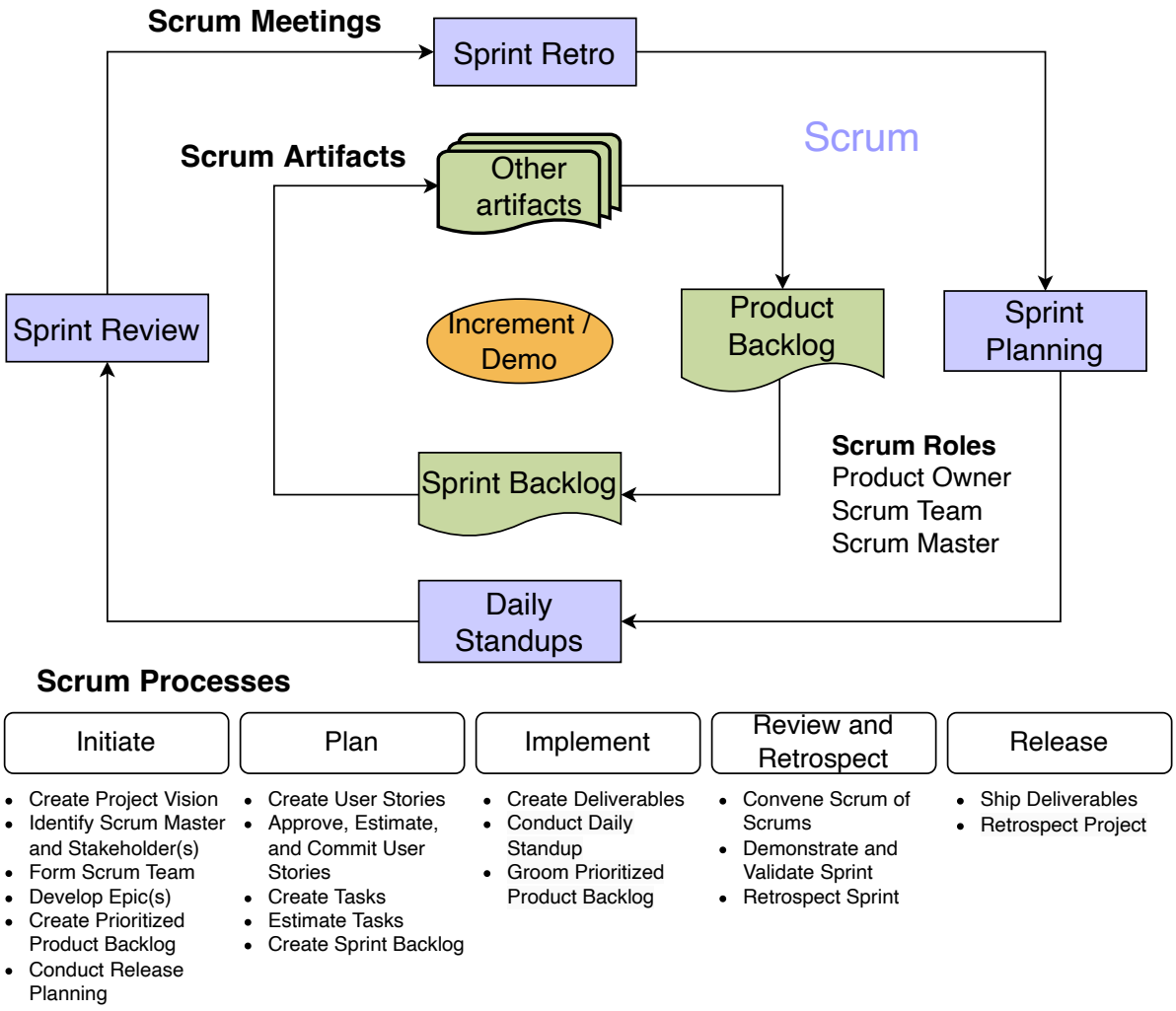


Figure 2.1: Scrum process overview.

- **Product Backlog**: a list of work items that includes features, functions, *User Stories*, and tasks needed to build a product [159]. With iterations, the *Product Backlog* would include enhancements, bug fixes, new product ideas, and deprioritized product features. The *Product Backlog* is developed and maintained in Scrum methodology using an Excel spreadsheet or a commercial Agile tool.

- **Sprint Backlog**: a subset of *Product Backlog* that includes work items prioritized to be developed during a Sprint [159]. The *Sprint Backlog* is created and maintained using an Excel spreadsheet or a commercial Agile tool.

- **User Story**: the smallest unit of work in the Scrum methodology. It is a general explanation of a software feature written from the customer's perspective. *User Stories* are listed in the sprint and *Product Backlog*s according to their priority. A user story is written in the following format [166]

  as a (role), I want (something), so that (benefit)

- **Product Roadmap**: this is a plan of action for how a product will be introduced and evolve over time and how it is broken down. The *Product Roadmap* is organized around *Epics* [25]. *Epics* are large bodies of work or *Capabilities* that span multiple iterations. *Epics* can be broken down into Use cases. Use cases can then be broken down into *User Stories*. The *Product Breakdown* is developed and maintained using an Excel spreadsheet, Word document, or a commercial Agile tool.

During the Scrum process (Figure 2.1), information elements are captured in an Agile tool, such as ClickUp [97], and amongst separate, independent documents (rather than a primary, integrated system model as in the Model-Based System Architecture Process (MBSAP) approach). According to the Guide to the Scrum Body of Knowledge (SBOK guide) [158], the Scrum processes are grouped into the five phases listed below.

- **Initiate**: This phase includes the processes related to initiating the system development. These processes include creating a project vision, identifying and forming the *Scrum Team*, identifying key stakeholder(s), developing *Epics*, creating a *Product Backlog*, and developing a *Release Plan*.

- **Plan and Estimate**: This phase consists of creating, estimating, approving, and commiting *User Stories* and tasks included in the *Sprint Backlog*.

- **Implement**: This phase is related to the execution of the tasks and activities to create the health tech system. These activities include providing the development team with the architecture artifacts to create the various deliverables. In this phase, the development conducts

Daily Standup meetings. The *Product Backlog* is regularly groomed at this phase. Grooming refers to reviewing and regularly updating the *Product Backlog Item (PBI)*.

- **Review and Retrospect**: This phase is concerned with reviewing the sprint deliverables and the work completed and brainstorming ways to improve the practices and methods used to execute the sprint.

- **Release**: This phase is about delivering the finally accepted deliverables to the customer. In this phase, identifying and documenting the lessons learned for the project occur.

Grouping the content of each viewpoint into a set of perspectives that create a logical and easily searchable content framework is discussed in Chapter 2.

2.2 Model-Based Systems Engineering (MBSE)

A model is an abstraction or representation of a system. The concept of utilizing models to control the complexity of software systems has been around for a while. In 1987, Ivar Jacobson introduced the concept of use cases [167]. He discussed how use cases were used at Ericsson to capture system *Requirements* using textual, structural, and visual modeling techniques to inform object-oriented analysis and design [168]. Basha et al. [169] argue that practitioners have primarily applied models to selected elements of the development process, particularly the *Structural* aspects in the design phase and model verification in the testing phase. The various stages of the software development lifecycle are insufficiently interconnected due to the lack of an integrated way to explain relevant concepts at a suitable level of abstraction [169].

MBSE is a software development approach in which models play an important central role [170]. MBSE was developed to overcome the drawbacks of the conventional Document-Based Systems Engineering (DBSE) method, which became apparent as information-intensive systems became more complex [171]. MBSE is based on the end-to-end use of formal, composable, and manipulable models in the Software life cycle. A developing direction is that modeling languages are domain-specific: they offer software developers the tools to capture essential characteristics

of their application domain [172]. MBSE is the idea of realizing code reusability and performing product development using software modeling techniques. The system described by a model may or may not exist when the model is created. Models are created to serve specific purposes, such as presenting a human-understandable system description [173, 174].

MBSE could be thought of as an attempt at having a knowledge hub for the software development lifecycle. The goal is to use models across the development process lifecycle to explain relevant concepts of each system view. Although MBSE has been around for many years, the generation of executable code from implementation-level models has attracted much interest over the last few years. It is important to note that MBSE encompasses a much broader scope and areas, such as business process modeling and enterprise-wide federated repositories [169]. MBSE suggests a role transition of software models from documentation to development [169]. Such transition requires completeness, precision, and model-implementation mapping, which is about generating model-prescribed code and managing the consistency between model and code over time [169]. MBSE in the context of complex software development is needed to describe the system from various views, such as *Structure*, *Behavior*, and non-functional properties [175, 176]. When not using model-prescribed code, the software developer should translate the model into code. In most cases, developers would instead write code directly [177], which emphasizes the importance of complete modeling [175].

MBSE offers practices to support the development of complex systems by reducing design errors, reducing rework, and improving system quality and overall project performance over traditional systems engineering methodologies [59, 178]. Some MBSE outcomes include enhancing communications by facilitating system presentation and understanding and enabling ongoing *Requirements* verification and validation. The advantages of these outcomes include more accurate estimation, improved system quality in terms of *Requirements Traceability*, reduced development risk, increased *Productivity*, and more standardized and effective capture of system knowledge [9].

Additionally, MBSE places a focus on systems engineering activities in the earlier stages of the system development life cycle to reduce the risk of accruing high costs associated with fixing

defects detected later in the project life cycle [179–181]. An MBSE approach helps manage system complexity during the architecting process, relative to traditional DBSE approaches, using a single model to represent key aspects of the system. An MBSE approach uses a set of languages, tools, and methods to facilitate the developing, navigating, communicating, and analyzing of an architecture model. MBSE languages, such as the Unified Modeling Language (UML), are used by business systems analysts and software architects to describe, specify, design, and document existing or non-existing business processes and *Structural* and *behavioral* views of software systems [182].

While DBSE uses documents and multiple tools to capture and communicate the system design throughout the project lifecycle. MBSE employs system architecture models to explain the relevant engineering aspects and relationships between these models [13] to communicate the system description. The shortcomings of DBSE practices became apparent as information-intensive systems became more complex. The shortcomings of DBSE include ambiguity due to informal language, slow update cycles that cannot keep up with a system's rapid evolution, and the lack of an effective way to perform an automated validation [12]. Due to these limitations, several US government agencies and commercial companies started to adopt MBSE as their systems engineering approach [171, 183, 184]. Better communication, managing complexity, improved product quality, and improved capturing of system knowledge have all been claimed as advantages of MBSE [185].

MBSE tools, such as Sparx Enterprise Architect, provide capabilities to visualize, analyze, model, test, and maintain all software, processes, and architectures. Sparx Enterprise Architect helps model and manage complex information by integrating and connecting a wide range of *Structural* and *Behavioral* information in a visual, coherent, and verifiable form [186]. MBSAP, an instance of MBSE, follows object-oriented design principles, such as abstraction, encapsulation, modularity, generalization, aggregation, interface definition, and polymorphism to architect the system through structured decomposition of its component into modular and manageable levels of complexity [1].

The discussion around MBSE is closely tied to the concept of architecture, which represents the *Structure*, *Behavior*, and *Rules* for a complex entity (or system), including its evolution over time [1]. The function of architecture modeling is to capture the key characteristics of a complex entity in a tool-supported structure [1]. An architecture framework, on the other hand, "establishes a common practice for creating, interpreting, analyzing, and using architecture descriptions with a particular domain of application or stakeholder community" [187]. Some commonly encountered architecture frameworks are shown in Figure 1.2.

There has been a tendency to use MBSE approaches due to the reported benefits. The International Council on Systems Engineering (INCOSE) describes the benefits of an MBSE approach as improved communications, increased ability to manage system complexity, improved product quality, and enhanced knowledge capture [188]. However, it is important to note that none of the MBSE methodologies [1, 189] shown in Figure 1.2 have gained the ever-increasing popularity of Scrum [152].

The concept behind model-based software engineering is to leverage software modeling to carry out development and maintenance and to achieve code reuse [10]. The notion of employing models to reduce software complexity has been around for many years. When appropriately used, MBSE can provide a significant opportunity to capitalize on individual intellectual assets in software engineering in general and to realize the promise of business/technology alignment made by Domain-Driven Design in particular [10]. However, MBSE can also pose a threat because of the additional challenges that it may introduce at the technical and organizational levels.

While adopting MBSE, several challenges have been identified and discussed by researchers and practitioners [171, 183, 184, 190]. Some of these challenges include the disconnect between how system architects conceptualize their systems (within the limitations of a typical DBSE approach) and describing them in a different way. Another challenge is related to the perception that MBSE is performed by a tool, although several researchers explained that MBSE is more than just a tool [171, 183, 191]. Usability is another issue that has been described, as too many aspects and attributes have to be specified to describe a simple system characteristic—in other words, too

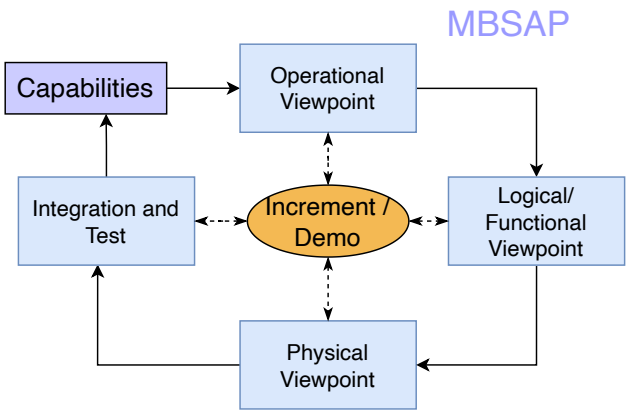


Figure 2.2: Overview of the MBSAP [1].

many clicks [12]. Finally, the selection of MBSE tools without a deep understanding of user needs is another issue [12]. Based on the review of 360 published articles, Henderson found very limited publicly available measured evidence of the potential benefits of adopting and implementing MBSE [192]. Accordingly, it is not a surprise that implementing an MBSE approach typically takes a long time and does not frequently provide incremental value to the customer. Therefore, searching for alternative approaches seems necessary.

2.3 Model-Based System Architecture Process (MBSAP)

The MBSAP outlines object-oriented design methods to create an architecture for a system through structured decomposition into modular and manageable levels of complexity by using object-oriented principles, such as abstraction, encapsulation, modularity, generalization, aggregation, interface definition, and polymorphism [1]. The process begins with identifying the customer's needs. Then, one iteratively develops progressive architecture models starting with an *Operational Viewpoint*, then a *Logical/Functional Viewpoint (LV/FV)*, and, finally, a *Physical Viewpoint (PV)*. Similarly, prototypes of the system (or system increments) are incrementally built, integrated, and tested with other increments. Finally, the cycle leads to either the delivery of a final product or a new starting point for a follow-on increment of development [1]. An overview of the MBSAP is shown in Figure 2.2.

Many practitioners of object-oriented methods make the assumption that the essence of an object-oriented method is the incremental approach [1]. This incremental (spiral approach) originally evolved to respond to frequently changing *Requirements*, especially for complex systems. It is obvious that the MBSAP is intrinsically incremental and iterative (as shown by the closed loop in Figure 2.2), making its integration with other Agile methods occurs naturally in an attempt to get the best of both approaches.

2.4 Towards Agile Model-Based Software Engineering

To address the gaps in the traditional systems development methods, in 2001, the Agile Manifesto [54] proposed 12 principles [193]. The Agile Manifesto served as a framework for developing methods, including Scrum, Extreme Programming (XP), Crystal, and Adaptive Adaptive Software Development (ASD) [194]. However, several issues and challenges were reported in the literature. Cho [194] studied the issues related to Scrum implementation and reported five main challenges: (1) lack of documentation led the developers who do not have much experience with the project or the system to consume more time, (2) lack of communication between teams led to several issues such as duplicate work, (3) lack of effective customer engagement, (4) lack of concentration due to open working space, and (5) inefficiencies in *Sprint Planning* and review meetings.

Several researchers argued that a combined Agile and traditional project management approach has merits. Boehm [44] and Baird et al. [164] suggested that a pure waterfall cannot meet the project objectives and that a mix of both approaches is needed. Lozo [195] argued that Agile and Waterfall could benefit from each other and advocate for a hybrid approach to managing information technology projects. Due to the benefits of both traditional and Agile project management methodologies, it is not possible to declare that one methodology is better than the other [196]. Finding the optimal mix would better contribute to the project's success [196]. Binder et al. [197] discussed the project management cocktail model and suggested an approach for balancing Agile and ISO 21500.

Agile MBSE presented itself as a possible solution for two issues that faced system development, namely, rigidity and waterfall orientation [18]. Agile MBSE also presented itself as a potential solution for the competing views and challenges related to documentation and *traceability*. The architecture specification document is usually very long, complex, and not self-explanatory [198]. Therefore, the Agile manifesto values "working software over comprehensive documentation" [54]. However, not all Agile practitioners seem to agree with this Agile principle [199]. Moreover, a survey conducted at the University of Melbourne revealed that despite the lower priority of documentation in Agile practices, 98% of the respondents considered documented information moderately to extremely important when estimating effort [200]. However, developers find documentation important, but at the same time, too little of it is available in their projects [199]. While some Agilistas devalue documentation and *traceability* [201], Agile methods and documentation are not actually contradictory [198, 202]. A certain amount of documentation is essential [198, 203]. Some researchers have also made a case that *traceability* is both necessary and required [25]. The view that supports documentation is adopted in government procurement/reporting practices [58]. Researchers see the need for techniques and methods that support documentation in Agile environments [27], such as a technique that makes documentation more easily writable, manageable, and updatable [28, 29].

Douglass [25] clarified that in Agile MBSE, the outcome of Agile software development is implementation, while the outcome of systems engineering is specification. Douglass [25] also discussed the notion of model-based handoff to "downstream engineering", enabling precise and unambiguous communication between architecture and *Requirements* analysts and the discipline-specific teams.

Salehi and Wang [22] compared four V-models and found that they did not consider the Agile concept. This finding led to the proposal of the adoption of Agile in MBSE to create a new approach, which was termed the Munich Agile MBSE Concept (MAGIC) [22]. Integrating MBSE and product development offers the capability of building a virtual prototype and a product's digital twin [23]. Bott and Mesmer [24] reviewed the theories supporting the Agile and MBSE methodolo-

gies and found them a key enabler of Agile methods for systems engineering. Figure 1.2 illustrates the relationship between Software Development Lifecycle (SDLC) models and MBSE.

There is also significant discussion on traditional DBSE using the waterfall and the V-model, which is considered an extension of the waterfall methodology [204]. Waterfall-based Systems Engineering methodology often does not meet schedule and cost objectives when designing, building, and deploying effective systems [33]. A study by Younse et al. [205] analyzed the benefits of an MBSE approach over a traditional Systems Engineering approach for architecting a robotic space system. Researchers also discuss the changes that need to happen to the V-model to cope with the needs of their system development efforts. Graessler et al. [206] discuss the modifications that need to be integrated into the V-Model to be prepared for future challenges. Liu et al. [207] discuss the Incremental V-Model Process for Automotive Development. Sell et al. [208] proposed a new approach for integrating the V-model and SysML for advanced mechatronics system design.

The Agility concept was created for software by the Agile Alliance but extended to Systems Engineering to improve product quality, improve engineering efficiency, and make solutions adaptable to ensure suitability and affordability [1]. The definition of Systems Engineering is centered around what it does; the how can be satisfied in many ways [209]. Similarly, Turner [210] agrees with Dove that Agile is a state of mind and philosophical approach rather than a set of rules. Haberfellner and Weck [211] distinguished between Agility in the Systems Engineering process versus Agility in the resulting system itself. Stelzmann [212] investigated the differences between the context of software and systems development (hardware systems). Moreover, existing literature and articles focused on technology innovation discuss the use of the Agile business model [161] and Agile software development [213] as the mainstream for tech startup software engineering projects [214].

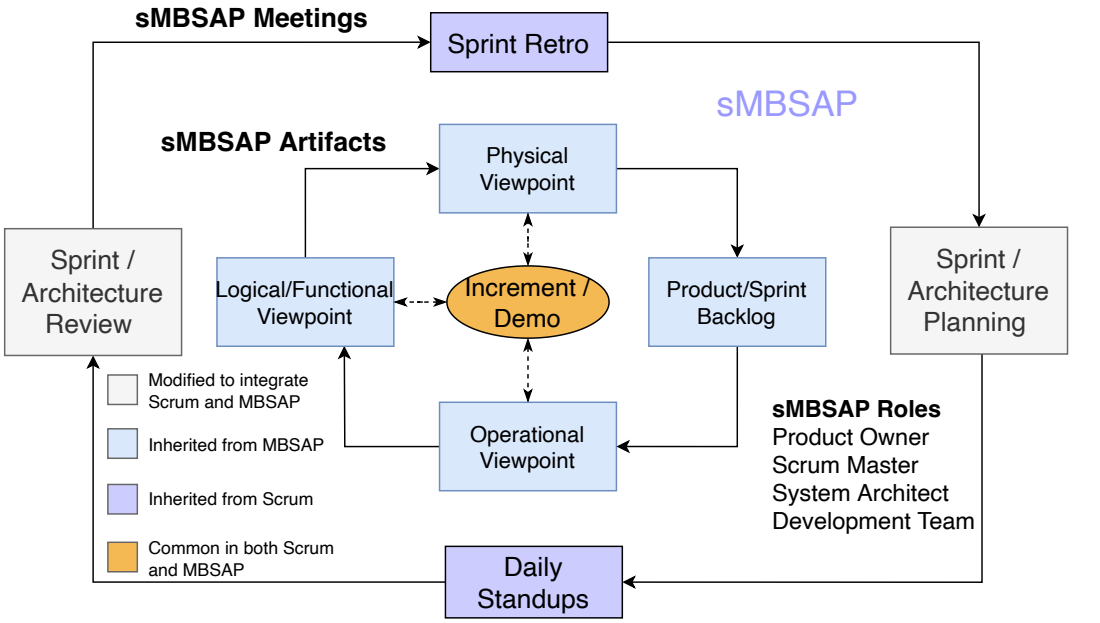
2.5 Integrated Scrum Model Based System Architecture Process (sMBSAP)

2.5.1 Overview of the sMBSAP

Before describing the sMBSAP method in detail, it is important to highlight that the illustrative development activity shown here is for an implemented health technology system, described in Section 1.3.3. The sMBSAP approach includes five phases: (1) Initiate, (2) Plan and Architect, (3) Implement, (4) Review and Retrospect, and (5) Release. The outputs of each phase serve as inputs to the following phase, as shown in Figure 2.3. Figure 2.3 also shows that two of the four main Scrum meetings are used during the sMBSAP approach, namely, *Daily Standups* and *Sprint Retro*. The other two Scrum meetings were modified for the sMBSAP. The *Sprint Planning* meeting was modified to be the *Sprint/Architecture Planning* meeting. The same applied to the *Sprint Review* meeting, which was modified to be the *Sprint/Architecture Review* meeting.

The typical MBSAP viewpoints generate the architecture artifacts for driving the development process. The MBSAP artifacts and *Sprint Backlog* that include *User Stories* are the key information that the *Product Development Team* uses to execute the product development and show the progress to stakeholders. The *Sprint Backlog* captures the list of items that need to be developed during each sMBSAP-driven sprint. The sMBSAP approach also includes the many typical MBSAP artifacts, including but not limited to a *Glossary*, *Product Breakdown*, *Class Diagrams*, *Object Diagrams*, *Data Models*, *Use Case Diagrams*, and *Capabilities* .

According to Borky and Bradley [1], the term "*Capabilities*" is a preferred term over the term "*Requirements*". *User Stories* are similar to *Requirements*, except they are written from the user's perspective—in other words, what a user shall do when using the system rather than describing what the system shall do for the user. The perspective of *Capabilities* captures the system *Capabilities* , which include *User Stories*, *Requirements*, and other behind-the-scenes tasks required to enable system *Capabilities* . Now, the following description of the phases and processes of the sMBSAP is organized to mirror that of the Scrum method for more straightforward mapping.



sMBSAP Processes

Initiate	Plan and Architect	Implement	Review and Retrospect	Release
Inputs • Project Business Case	**Inputs** • Personas • Prioritized Product Backlog • System Architecture Overview and Summary • Release Plan • Core Project Team	**Inputs** • Acceptance criteria of the Product Backlog Items • Sprint Backlog • System Architecture	**Inputs** • Prioritized Product Backlog • Updated System Architecture • Sprint Deliverables • Burndown Charts	**Inputs** • Accepted System Architecture • Accepted Deliverables • List of Actionable Improvements
Processes • Create Project and Product Scope • Identify Project Stakeholders and Project Team • Create Architecture Overview and Summary • Create Product Breakdown • Create Product Backlog • Develop Release Plan	**Processes** • Create and Update Backlog Items • Develop System Architecture • Commit User Stories • Create Product Breakdown • Estimate Backlog Items	**Processes** • Create Deliverables/Product Increments • Communicate Progress • Groom Product Backlog Breakdown • Update System Architecture	**Processes** • Demonstrate and Validate Deliverables • Retrospect Sprint	**Processes** • Ship Deliverables • Deliver Architecture Models • Retrospect Project
Outputs • Personas • Prioritized Product Backlog • System Architecture Overview and Summary • Release Plan • Core Project Team	**Outputs** • Acceptance criteria of the Product Backlog Items • Sprint Backlog • System Architecture	**Outputs** • Prioritized Product Backlog • Updated System Architecture • Sprint Deliverables • Burndown Charts	**Outputs** • Accepted System Architecture • Accepted Deliverables • List of Actionable Improvements	**Outputs** • Working Deliverables • Lessons Learned for Future Implementation

Input, process or output ⇨ Inherited from MBSAP
Input, process or output ⇨ Inherited from Scrum

Figure 2.3: Overview of the sMBSAP with the goal of developing *Product Increments*.

2.5.2 Initiate

This phase includes the processes related to initiating the project. These processes are summarized below:

- Create Project and Product Scope: In this process, the project business case is reviewed to create a *Project Scope Statement* and subsequent *Product Scope*. The *Product Owner* is typically identified at this stage of the project.

- Identify Project Stakeholders and Project Team: In this process, the four project roles are identified, which include the *Product Owner*, *Scrum Master*, *System Architect*, and *Product Development Team*. Other project stakeholders are also identified during this process.

- Create *Architecture Overview* and Summary: In this process, an *Architecture Overview* that provides the following architecture-related information is created: the architecture's scope, purpose, and perspective, contextual information, the role of the *System Architect*, and the timeline of the architecture's development. The *Architecture Overview* acts as a contract between stakeholders and the *System Architect* based on establishing bilateral commitments and understanding of the role of the architecture effort within the overall project effort [1]. The main customer for the architecture artifacts is the *Product Development Team*; accordingly, the *System Architect* must explain the value and contribution of the architecture process. The *System Architect* should also expect organizational resistance and lack of support among software developers, especially those who are used to writing code with very little or no documentation as input. In this process, the *System Architect* decides which MBSE tool they will use throughout the project.

- Create *Product Breakdown*: In this process, the *Project Scope Statement* and *Product Scope* are used as the basis for breaking the product down into *Epics*, *Use Cases*, *User Stories*, and *Requirements*, as shown in Figure 2.4. The *Product Breakdown* is an iterative process that occurs first during the Initiate phase and is further refined through meetings between the project team and key stakeholders in subsequent phases as needed. In the sMBSAP

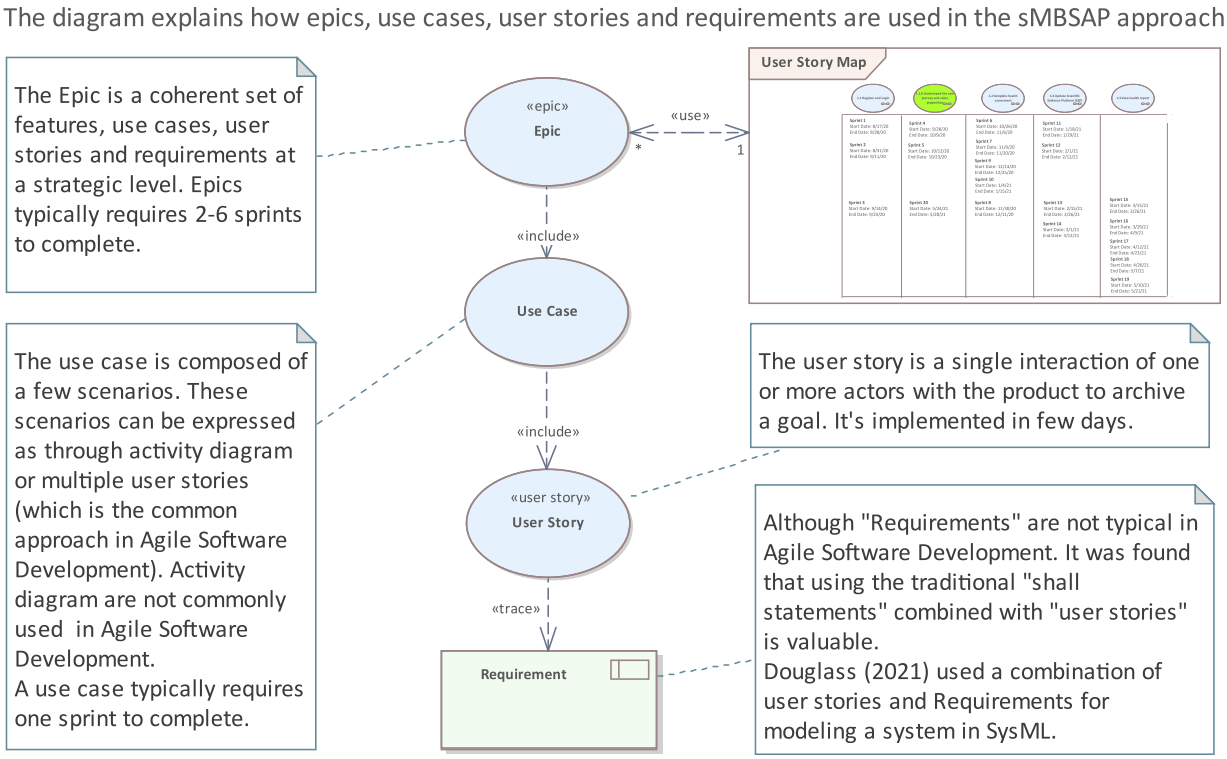


Figure 2.4: Relationship among *Epics*, *Use Cases*, *User Stories*, and *Requirements* in the sMBSAP method.

approach, *Epics* are modeled as stereotyped *Use Cases* [25] and are decomposed to (lower-level) *Use Cases*, which are, in turn, decomposed into *User Stories*, which are broken down into *Requirements*. This taxonomy is shown in Figure 2.4.

- Create *Product Backlog*: In this process, *User Stories*, *Requirements*, and other tasks are added to the *Product Backlog*. These items are referred to as *PBIs*. It is important to note that a project team may choose to use only *User Stories* (commonly used in Scrum) or both *User Stories* and *Requirements* (*Requirements* are commonly used in MBSE and systems engineering). The *PBIs* will be progressively refined, elaborated, and later prioritized. The acceptance criteria are also established at this point and will be further elaborated. The *Product Backlog* is developed and maintained by using a*Requirement* management tool or Agile development tool, such as Clickup [163]. Alternatively (or in addition), *User Stories* and *Requirements* may be visually captured in the model, as shown in Figure 2.5, which illustrates *User Stories* and *Requirements* that are modeled as stereotyped *Use Cases*, and

Requirements are traced to *User Stories*. This allows a *User Story* to be described and to its connection to a persona to be shown. In this *Use Case* diagram, the *System Architect* wants to capture the interaction of the "User" with the "Health Assessment" module of the system and communicate it with the *Product Development Team*. The *System Architect* explains the "User" *Behavior* through a combination of *User Stories* and *Requirements*. At the beginning of the "Health Assessment", the system displays a series of messages to the "User" to allow them to customize their "Health Assessment" experience. The "User" will specifically be asked to select whether they would like to receive one question per page, to select the weight and height unit of measure, to select which health assessment sections to complete, and whether the "User" prefers to focus on a specific category of medical conditions. The "User" will then start navigating the "Health Assessment" sections. The "User" will have the ability to transition from one section to the other. They can also skip questions and come back to them later. The system will display a message at the end of each section to transition the "User" from one section to the other. During the navigation, the "User" can see their progress in terms of the percentage of completion. If the "User" does not complete the "Health Assessment", the system will send weekly emails reminding the "User" to complete the "Health Assessment".

- Develop Release Plan: In this process, the Product team develops a *Release Plan*, which is basically a phased deployment timeline that can be shared with the project's stakeholders. The length of each sprint is usually decided in this process. Some Scrum practitioners develop a *Product Roadmap* that is more strategic and high-level and a *Release Plan* that is more tactical and detailed.

2.5.3 Plan and Architect

This second phase consists of the processes related to planning, architecting, and estimating the *PBIs*. These processes are summarized below. It is important to note that although these processes

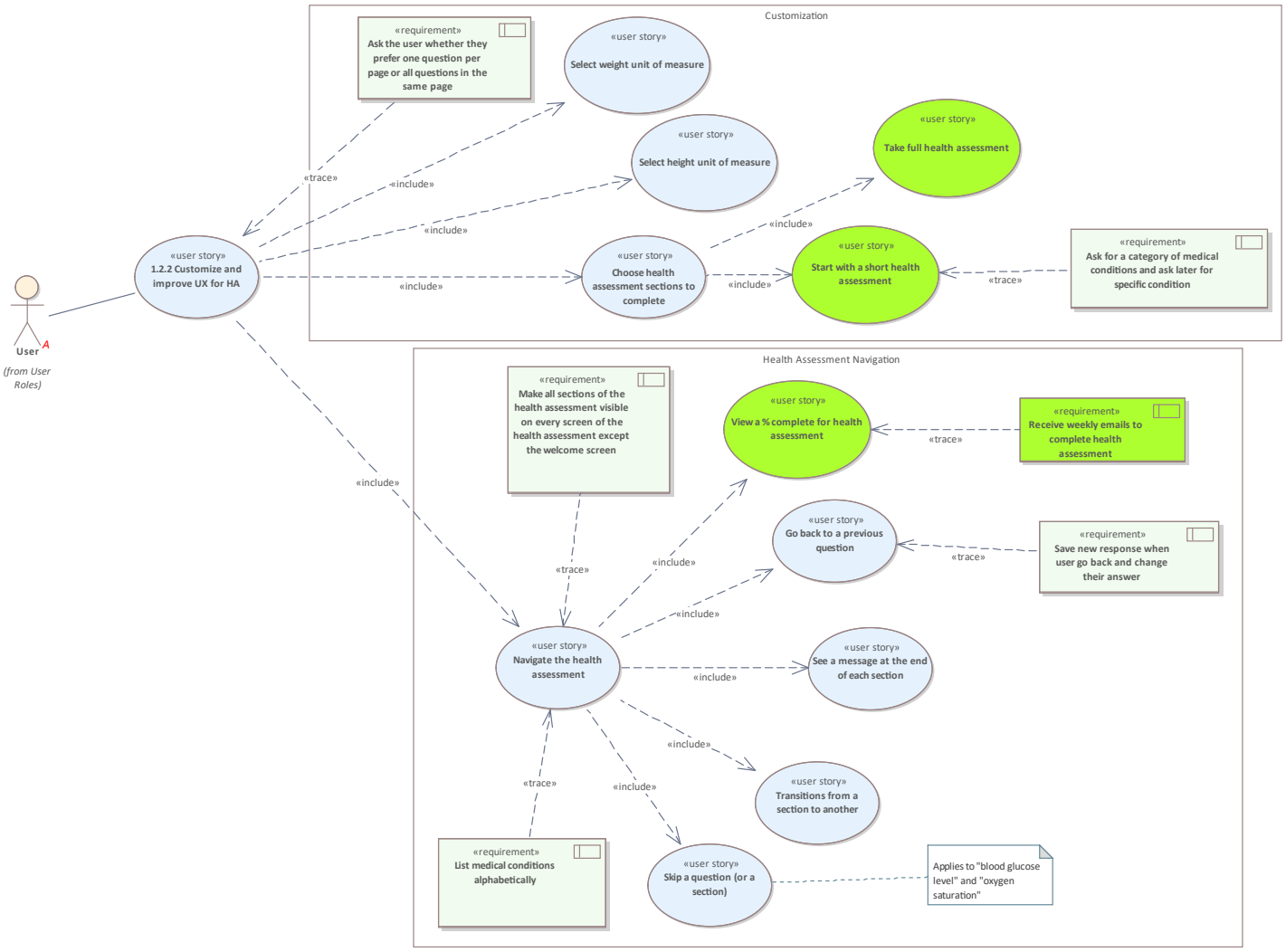


Figure 2.5: An example of *Use Cases* stereotyped as *User Stories* and *Requirements*.

are presented sequentially, in practice, they overlap, and the outcome of later processes serves as an input for former processes.

- Create and Update Backlog Items: In this process, *PBIs* (*User Stories*, *Requirements*, and tasks) and their related acceptance criteria are created or updated and incorporated into the *Product Backlog*. *User Stories* are designed to ensure that the project *Requirements* are clearly defined and can be entirely understood by all stakeholders. When a *User story* is committed, it can be broken down into specific tasks (or *Requirements* and tasks). Agile development tools can show the task list beneath the relevant *User Story*.

- Develop System Architecture: In this process, the *System Architect* progressively develops the system architecture. The system architecture is developed in lockstep with the *User Stories*. Both are used by the *Product Development Team* to execute the development work. The three main viewpoints progressively developed during this process are:

  – Operational Viewpoint (OV): The first progression is concerned with translating the *Project Scope Statement*, *Product Scope*, and *PBIs* (in any form that they are expressed) into an architectural model known as an *Operational Viewpoint (OV)*. This mapping is achieved with *Use Cases* and other object-oriented constructs. Several researchers, such as Lattanze [215], stressed the importance of starting with a high-level view of the architecture before progressing to a more detailed design. The high-level view of the architecture is the primary purpose of the OV. The OV also defines the system's boundary and context. It also creates top-level partitioning (*Domains*), primary *Behaviors* (*Use Cases*), and primary data content. With the aid of the model, the *System Architect* maps *User Stories* to *Domains* and *Use Cases*. The data model developed in this first progression is called "Conceptual Data Model (CDM)". This is the most abstract type of data model. Platform-specific information, such as data types, sequences, procedures, and triggers, are not included in the CDM. Because of its simplicity, it is useful for communicating ideas among different stakeholders. *Data Models* can be de-

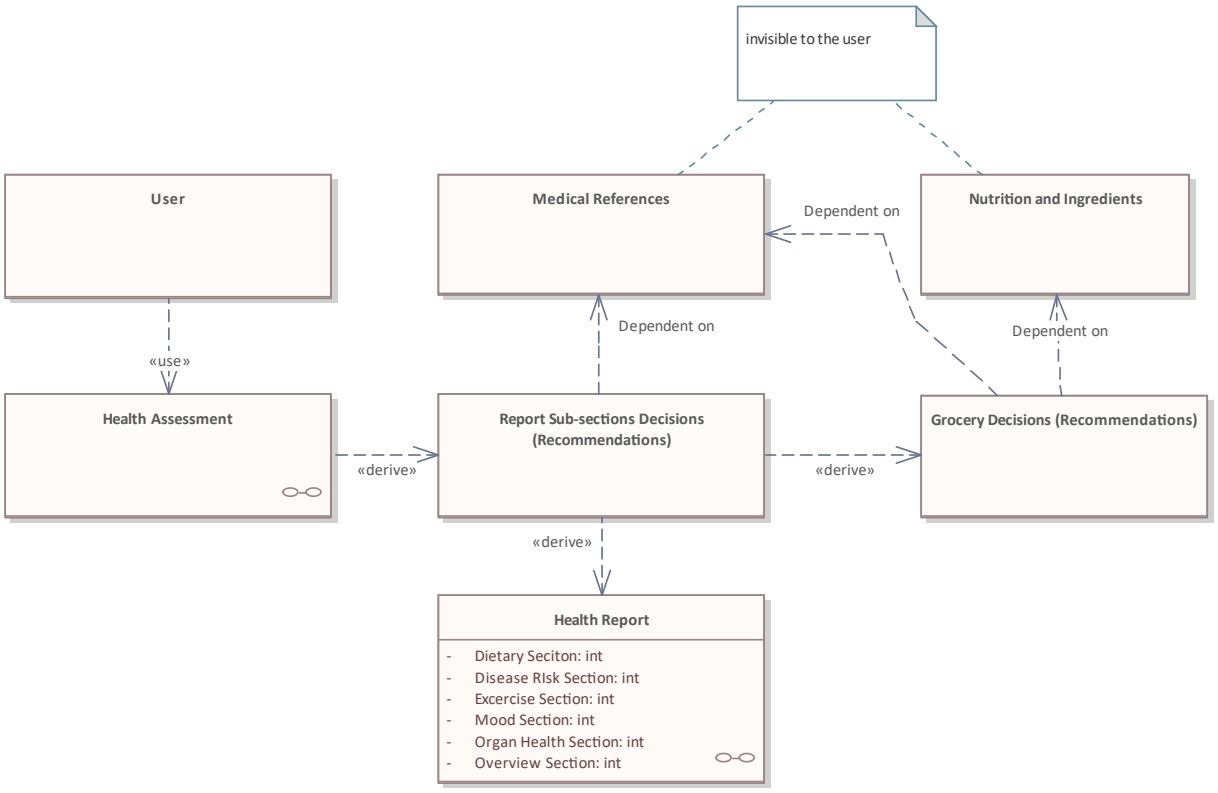


Figure 2.6: Conceptual Data Model (CDM) for a health technology system.

veloped with a number of notations, such as Information Engineering, IDEF1X, UML data modeling, and Entity Relationship notation.

The conceptual UML-based data model developed for the health tech system is shown in Figure 2.6. At this stage, the *System Architect* wants to capture and communicate the types of data (or "*Entities*") that the health tech system needs with the *Product Development Team*. These *Entities* include the "User", "Health Assessment", "Report Subsections Decisions", "Medical Reference", and others. In addition to *Entities*, the CDM also captures the *Relationships*, i.e., how an *Entity* connects to other *Entities*. In the case of the health tech system, the "User" takes the "Health Assessment". Based on the results of the "Health Assessment", the "Report Subsections Decisions" will be displayed to the "User" and form the basis of the "Health Report". The "Report Subsections Decisions" rely on the "Medical Reference" for communicating the

recommendations to the "User". The "Grocery Recommendations" are derived from "Report Subsections Decisions" and depend on both the "Nutrition and Ingredients" and "Medical Reference". As noted on the CDM, both "Nutrition and Ingredients" and "Medical Reference" are not exposed to the "User".

– Logical/Functional Viewpoint (LV/FV): The next progression transforms the OV into the *Logical/Functional Viewpoint (LV/FV)*. This is where the design begins, using UML *Class Diagrams*, to define the details of system elements, functions, and data. The LV is a progressive elaboration on the perspectives of the OV and is molded mainly by decomposing *Domains* and *Use Cases* to develop *Structural* and *Behavioral* diagrams. The functional service specifications are developed and allocated to logical components and interfaces. The architectural layering is defined. The LV represents a functional definition of the technology- and product-agnostic system. The data model developed in this architecture iteration is called a "Logical Data Model (LDM)". The LDM defines the detailed *Structure* of a system's data elements and the *Relationships* between them. It elaborates on the CDM introduced during the OV progression, but without going to the level of specifying the Database Management System (DBMS) that will be used. LDM forms the basis of the "Physical Data Model (PDM)". This model is commonly developed by using the UML Data Modeling notation. The logical UML-based data model developed for the health tech system is shown in Figure 2.7. As shown in Figure 2.7, the data elements "Medical References" and "Nutrition and Ingredients" contain UML attributes; the names and generic data types remain platform-independent. Platform-specific data types and other metadata that relate to a specific DBMS implementation are defined by the PDM.

– Physical Viewpoint (PV): The architecture modeling is completed by progressing from the LV to the *Physical Viewpoint (PV)*. The PV is the basis for the actual implementation of the full system or an increment of the system. To clarify the relationship between the LV and the PV, the former defines what is to be built, and the latter de-

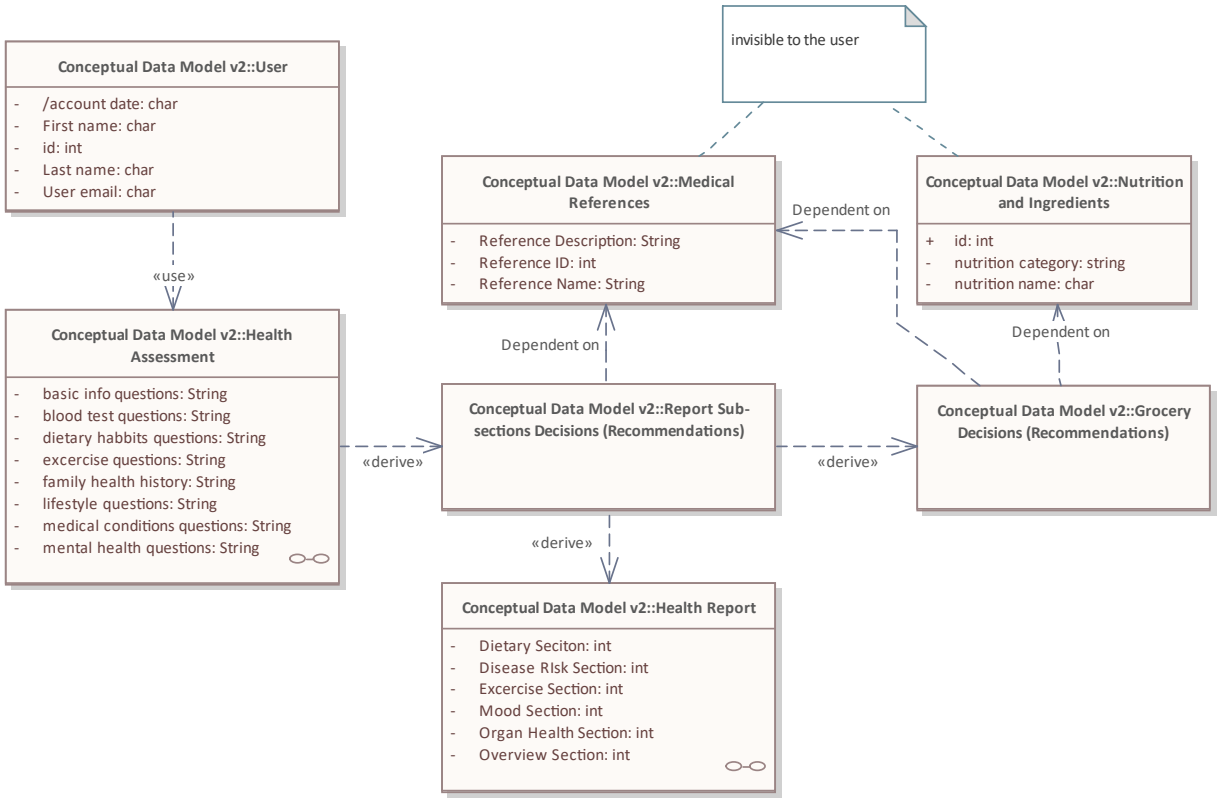


Figure 2.7: Logical Data Model (LDM) for a health technology system.

fines how it will be realized [1]. Accordingly, this architecture iteration focuses on products and standards whose selection is paramount to reaching a physical design. The data model developed during the PV is called a "Physical Data Model (PDM)". A PDM graphically represents the *Structure* of data as implemented by a relational database schema. The ability to automatically generate the database schema from a PDM is a significant advantage of PDMs, in addition to presenting a visual abstraction of the database structure. This is made feasible by the amount of metadata that a PDM captures and its close alignment with aspects of the database schema, such as database tables, columns, primary keys, and foreign keys. The UML-based PDM developed for the health tech system is shown in Figure 2.8. Each table is represented by a UML Class; table columns, primary keys, and foreign keys are modeled by using UML attributes and operations. The DBMS type used in the system is PostgreSQL.

It is important to note that each viewpoint is represented with several perspectives (within the viewpoints); the perspectives are largely derived from the fundamental elements of the architecture and the needs of the project. The proposed perspectives for the sMBSAP are the *Structure*, *Behavior*, *Data*, and *Requirements*, as shown in Figure 2.9. The importance of an adequate model *Structure* in achieving the full benefits of MBSE should be emphasized [1]. One way to group the content of each viewpoint into a set of perspectives that create a logical and easily searchable content framework is shown in Figure 2.10.

- Commit User Stories: In this process, the project team commits to delivering the approved *User Stories* for a sprint. The committed *User Stories* are added to the *Sprint Backlog*. During the *Sprint/Architecture Planning*, the *Scrum Team* may add further details to the *PBIs*.

- Estimate Backlog Items: In this process, the project team, supported by the *System Architect*, estimates the *PBIs* and estimates the effort required to develop the functionality described in each *PBI*.

2.5.4 Implement

This third phase is related to the execution of the activities required to develop the *Capabilities* described by the *PBIs* to create the product. These processes are summarized below.

- Create Deliverables/Product Increments:

  In this process, the project team works on the items in the *Sprint Backlog* to create sprint deliverables. The project team's progress, measured in completed story points, is captured in an Agile development tool. Planned versus actual story points are captured in the tool, in addition to marking an item as "done" when it is completed. The collected data are plotted on burnup charts and velocity fluctuation charts to allow the *Scrum Master* to monitor the project's health and make course corrections when needed.

It is important to note that creating deliverables/*Product Increments* may include activities such as project management, software engineering, continuous integration and testing, system configuration management, security, and other aspects. The sMBSAP is similar to the MBSAP in that the design modeled in the PV is built up in a prototype and goes through continuous integration and testing to assess its suitability against the required *Capabilities* that are being addressed.

- Communicate Progress: In this process, the project team members update each other on their individual progress and any barriers that they may be facing. These updates occur through a short daily 15 min meeting, referred to as a *Standup Meeting*. The *System Architect* participates in these meetings and addresses any issues that the *Product Development Team* faces in relation to the system architecture.

- Groom Product Backlog: In this process, the prioritized *PBIs* are continuously updated and refined. A backlog grooming meeting is conducted to discuss any changes or updates to the backlog.

- Update System Architecture: In this process, the system architecture models are continuously updated and refined based on the progress and feedback from the project team. The results of the architecture changes or updates are discussed during the *Sprint/Architecture Review*.

2.5.5 Review and Retrospect

This fourth phase is concerned with reviewing the deliverables and work completed and identifying areas for improvement for future consideration. The processes of this phase are summarized below:

- Demonstrate and Validate Deliverables: In this process, the *System Architect* presents the updated system architecture to the project stakeholders. The project team then demonstrates the sprint deliverables that match the models to the stakeholders. These presentations and

Figure 2.8: Physical Data Model (PDM) for a health technology system.

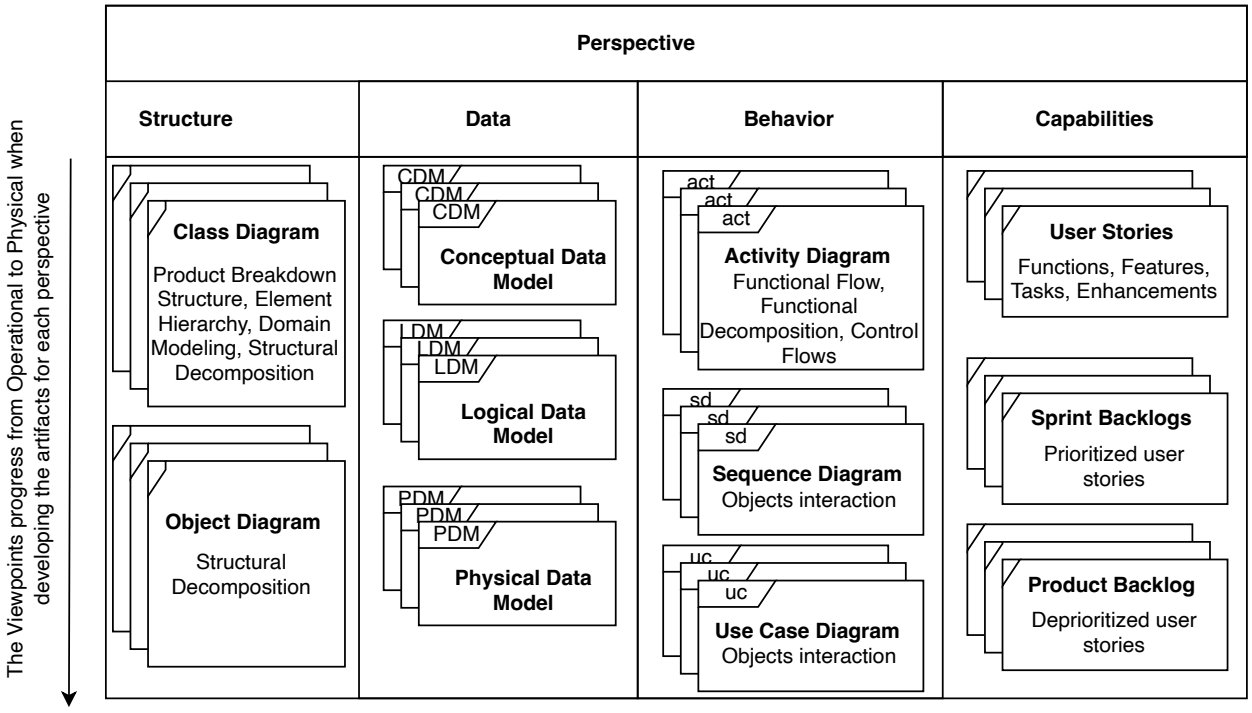


Figure 2.9: Organizational overview of an information model for the UML-based sMBSAP.

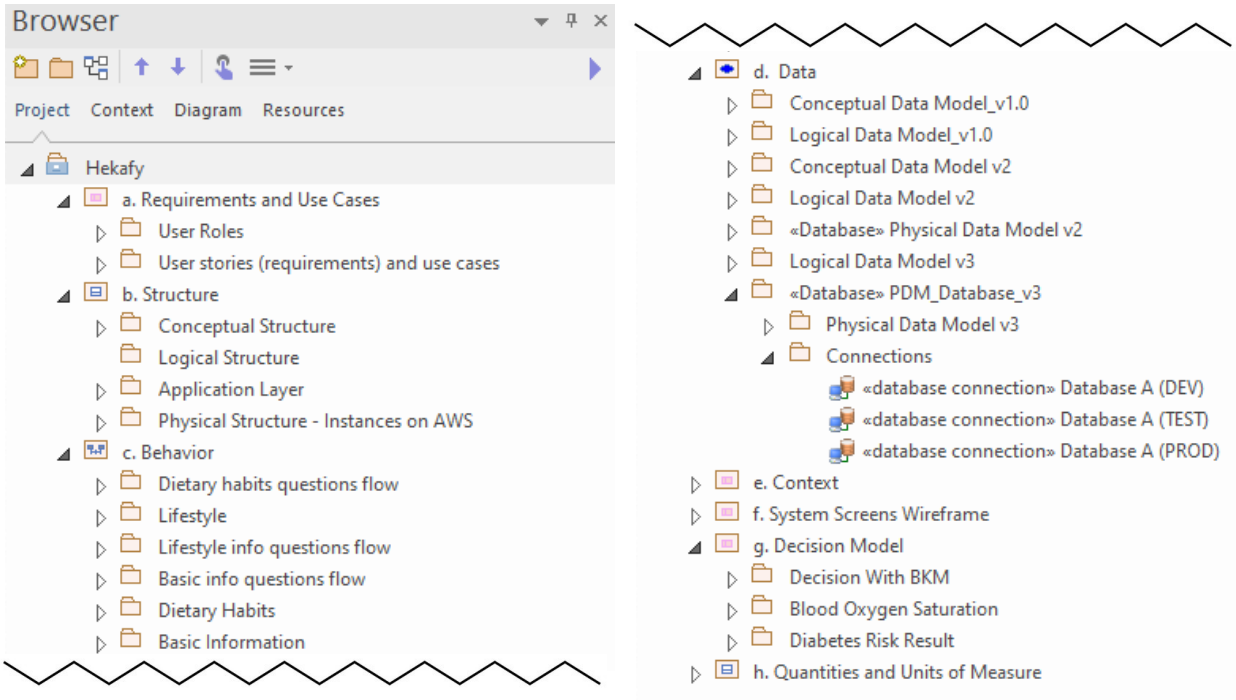


Figure 2.10: An illustration of organizing the sMBSAP artifacts in an MBSE tool.

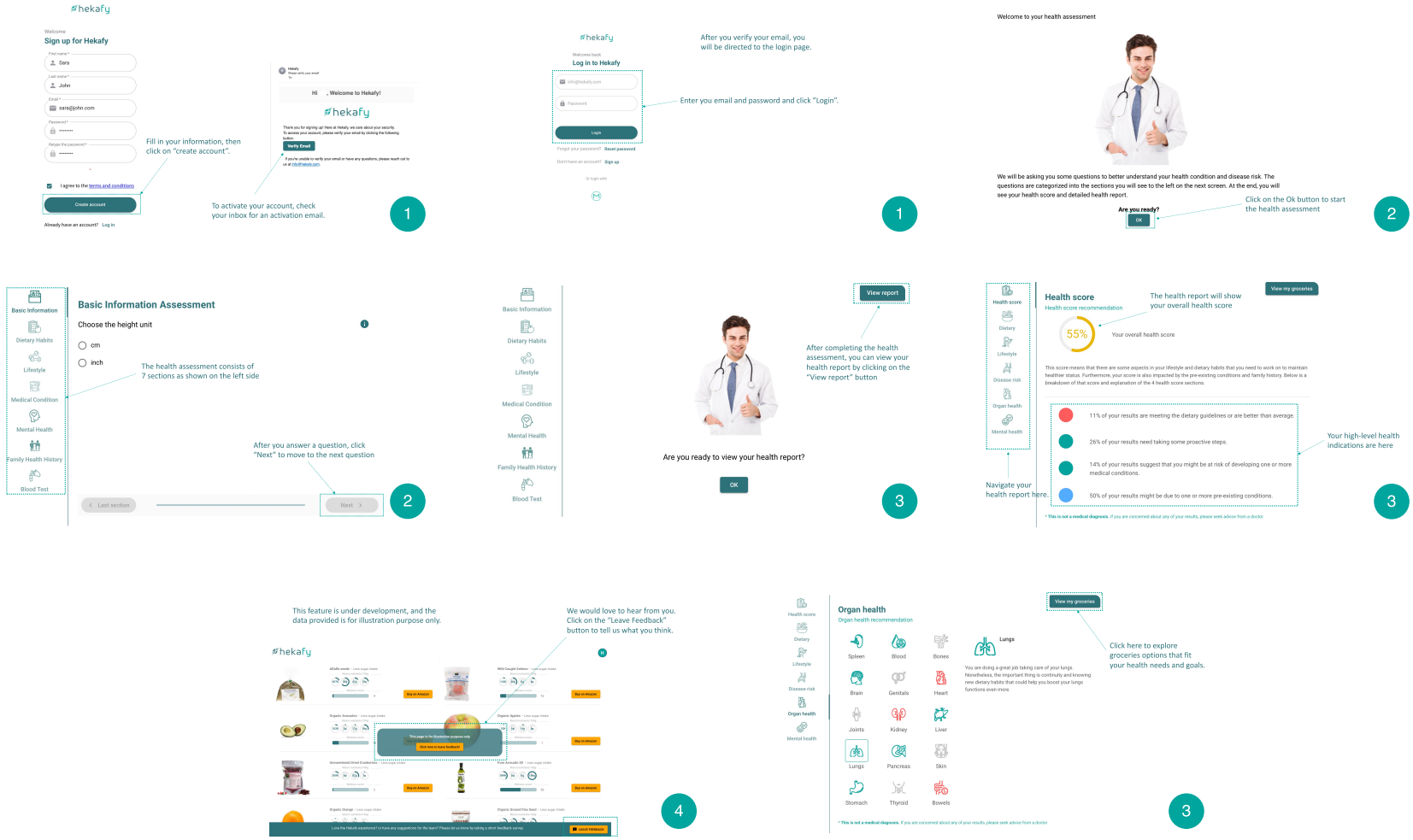


Figure 2.11: Screenshots from the health tech product demo.

demos occur in a *Sprint/Architecture Review* meeting. This meeting aims to gain the acceptance of the delivered *PBIs* from the *Product Owner*.

Screenshots from the health tech system product demo are shown in Figure 2.11. The product demo shows the four key steps in the "User" journey at a high level. In step 1, the "User" completes the registration process by entering their first name, last name, and email address, creating a password, and confirming it. After the "User" confirms their email address, they are redirected to the login page. In step 2, the "User" will be introduced to the "Health Assessment" and start completing its various sections. At the end of the "Health Assessment", the "User" will proceed to the third step, which is reviewing their "Health Report". Finally, in the fourth step, the "User" will receive the grocery items recommended for their health profile.

- Retrospect Sprint: In this process, the project team meets in a *Sprint Retro* meeting to discuss the lessons learned from the previous sprint. This information is recorded and should be used in future sprints. As a result of this meeting, some actions to improve performance or to make course corrections may be decided upon.

2.5.6 Release

This fifth and final phase is about delivering the finally accepted deliverables to the customer. In addition, the lessons learned from the project are identified and documented. These processes are summarized below.

- Ship Deliverables and Architecture Models: In this process, approved and accepted deliverables are handed over to the concerned stakeholders. A formal transition document should be drafted and signed by the concerned stakeholders denoting the successful completion of the agreed-upon shippable part of the product. The architecture models are also handed over to the concerned stakeholders. The combined artifacts developed for the health tech system are shown in Figure 2.12.

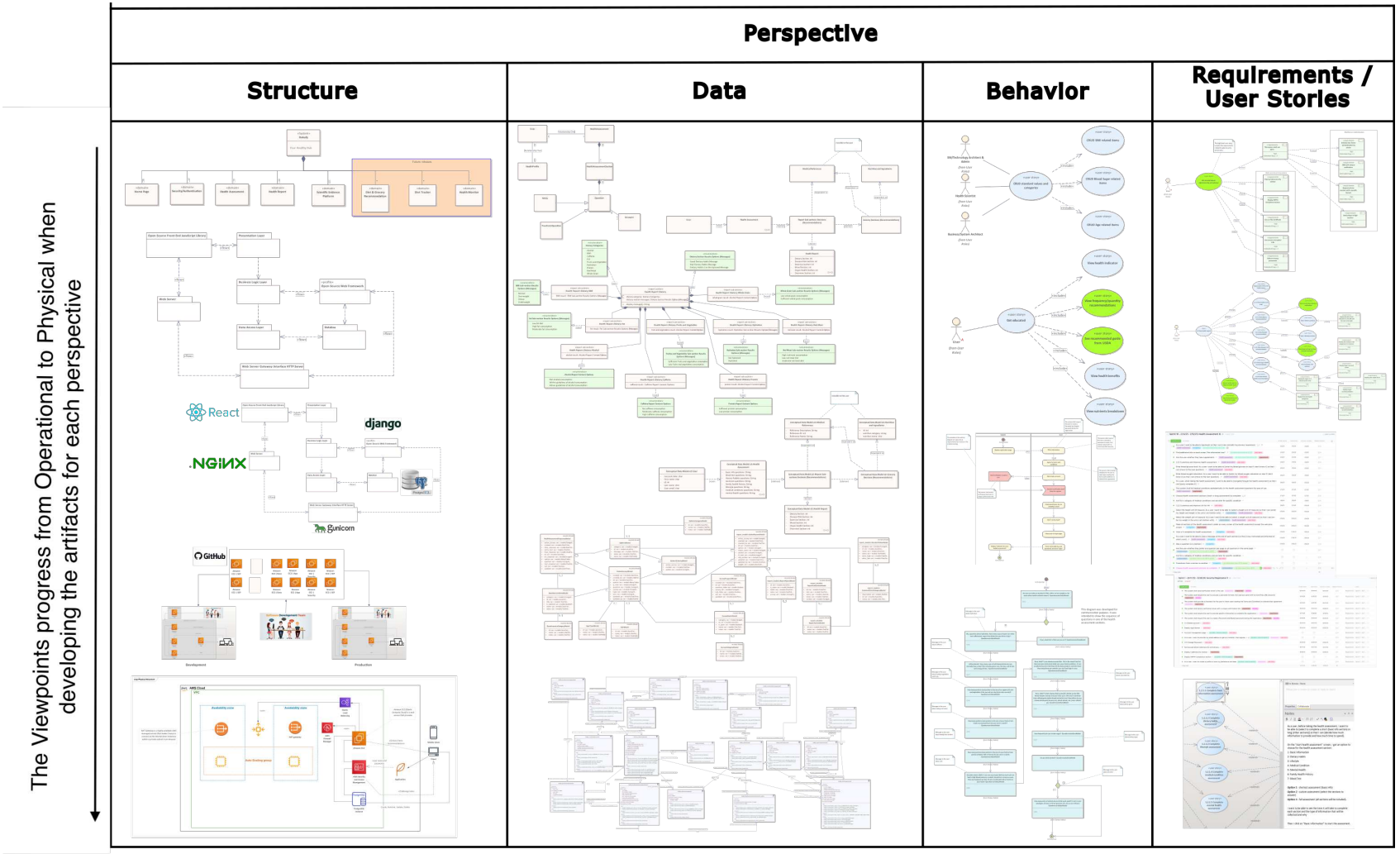


Figure 2.12: Combined sMBSAP artifacts for a health tech system.

- Retrospect Project: This is the final step in the project, in which the project team and stakeholders meet to identify and document the lessons learned for future implementation. This meeting is called the *Project Retrospective* meeting (or *Retro*).

2.6 sMBSAP Conclusion

This chapter introduces the proposed Scrum Model Based System Architecture Process (sMBSAP) approach. It provides a background about the main methodologies used to develop the sMBSAP, which includes Scrum, MBSE, and MBSAP. The chapter also presents a literature review surrounding the notion of integrating Agile and MBSE. The remainder of the chapter was dedicated to providing details of the building blocks of sMBSAP, including processes, including meetings, artifacts, processes, and roles. The sMBSAP approach includes five main artifacts: *Product/Sprint Backlog*, *OV*, *LV/FV*, *PV*, and *Product Increment*, as well as four roles: *Product Owner*, *Scrum Master*, *System Architect*, and *Product Development Team*. The sMBSAP updates the artifacts to keep system information within the model. The five sMBSAP phases include Initiate, Plan and Architect, Implement, Review and Retrospect, and Release. sMBSAP is application-agnostic as it can be applied to other software or engineered systems.

Chapter 3

Research Methodology[3]

Existing literature suggests that using a combined Agile Model-Based Systems Engineering (MBSE) approach provides benefits that could not be realized by either approach alone. This research adopts a specific instance of the Agile approach (i.e., Scrum) in combination with a specific instance of an MBSE approach (i.e., Model-Based System Architecture Process (MBSAP) [1]) into an Agile MBSE approach called the integrated *Scrum Model Based System Architecture Process (sMBSAP)*. The purpose of this research is to compare two software engineering approaches, the Scrum software development process and *sMBSAP*.

It would be useful to remind the reader of the overarching research question being investigated in this research project, that is:

Comparing sMBSAP and Scrum, are there measurable benefits to software development performance when using one approach over the other?

The overarching research question is translated into three hypotheses. Each hypothesis is concerned with one system performance objective (dependent variable). The first hypothesis corresponds to *Reliability of Estimation* (Hypotheses H_0 and H_1). The second hypothesis corresponds to *Productivity* (Hypotheses H_0 and H_1). The third hypothesis corresponds to *Defect Rate* (Hypotheses H_0 and H_1). In order to investigate the hypotheses, a quasi-experiment was conducted to architect and develop a prototype health technology system. Section 1.3 provides an overview of the health technology system. This study is an attempt to fill the gap in the existing literature and assess whether Agile MBSE has quantifiable benefits that may not be achieved by the Scrum approach alone.

This chapter presents the research methodology, including philosophy, approach, design, techniques, procedures, and quality of research design. The chapter is structured into five main sec-

[3]Parts of this chapter previously appeared in [2, 3].

tions; the first Section 3.1 provides fundamental background about the research philosophy and approach and an overview of the research methodology choices. The second section 3.2 explains in detail the research design, including methodological choice (3.2.1, research strategy (3.2.2), experimental design (3.2.3), experimental setting (3.2.4) and time horizon (3.2.5). The third section 3.3 clarifies why the quasi-experimental design was selected and how the present experiment was conducted. It then explains how the data for the study was obtained, covering the materials and instruments along with the data processing and analysis tools used. This section explains the four main activities and procedures conducted to carry out the quasi-experiment. The fourth section 3.4 discusses the procedures used by the researcher to counteract potential threats to research quality. The section starts with providing a background of reliability and validity (3.4.1). Then, continues to discuss the factors put into the researcher's consideration to mitigating threats to reliability (3.4.2), reliability, mitigating threats to validity (3.4.3), and finally mitigating threats to the statistical conclusion (3.4.4). The fifth section 3.5 provides a summary of this research methodology chapter.

3.1 Research Philosophy and Approach

Research can be designed to fulfill either an exploratory, descriptive, explanatory, or evaluative purpose or some combination of these [216]. Research Philosophy is an overarching term related to the development of knowledge and the nature of that knowledge. Saunders et al. [216] identifies four research philosophies: *positivism*, *realism*, *interpretivism*, and *pragmatism*. In the *positivism* philosophy, only phenomena that the researcher can observe will lead to the production of credible data. The researchers use existing theory to develop hypotheses that will be tested and confirmed, in whole or part, or refuted, leading to the further development of theory, which may then be tested by further research [216]. This type of research philosophy is usually considered quantitative as the problem is identified from literature hence only focusing on certain variables. This research project reflects the philosophy of positivism.

Several researchers discussed the *deduction*, and *abduction* approaches to theory development [216–219]. This research project followed the *deductive* approach in which the researcher hypothesized that an integrated Agile MBSE approach has an impact on software development performance compared to Agile alone. The researcher then designed a research strategy to test the hypothesis using three dependent variables: *Estimation Reliability*, *Productivity*, and *Defect Rate* data.

3.2 Research Design

The first section (Section 3.1) briefly introduced research philosophy, and approach to theory development used in this research project. This section intends to uncover the next stage, research design. The research design includes three sub-stages: methodological choice, research strategy, and choosing the time horizon for the research. These three sub-stages are followed to turn the research questions into a research project [216].

3.2.1 Methodological Choice

Several researchers discuss three common research methods, *qualitative*, *quantitative*, and *mixed methods* [220–224] The design of this research falls under the quantitative research design category. *Quantitative* research designs are generally associated with *positivism*, especially when used with predetermined and highly structured data collection techniques [216]. Some researchers advocate that there is an exclusive link between *positivism*, *deduction*, and a *quantitative* research design [225, 226]. *Quantitative* research is usually associated with a *deductive* approach, where data are collected and analyzed to test a theory. However, it may also incorporate an *inductive* approach, where data are used to develop theory [227].

Generally, a *quantitative* research design may use a single data collection technique, known as a *mono method quantitative study*. A *quantitative* research design may also use more than one *quantitative* data collection technique, known as a *multi-method quantitative* technique followed

by an analytical procedure [216]. This research project is a *mono method quantitative* study in which data were collected using a single method, which is documentation, from multiple sources.

Saunders et al. [216] explain that quantitative research is primarily associated with experimental research strategy (the strategy selected for this research) and summarize the characteristics of quantitative research in the following points which apply to this research project:

- Designed to examine relationships between variables
- Often uses probability sampling techniques to ensure generalisability
- Method(s) used to collect data are rigorously defined and highly structured
- Collection results in numerical and standardized data
- Analysis conducted through the use of statistics and diagrams
- Resulting meanings derived from numbers

3.2.2 Research Strategy

In general terms, a strategy could be defined as an adaptation that serves an important function in achieving evolutionary success [228]. A research strategy may therefore be defined as a plan of how a researcher will go about answering their research question [216]. It is the methodological link between research philosophy and subsequent methodological choice to collect and analyze data [229]. Saunders et al. [216] discuss eight research strategies: *experiment*, *survey*, *archival and documentary research*, *case study*, *ethnography*, *action research*, *grounded theory*, and *narrative inquiry*.

This study used an experimental research design to address the overarching research question and hypotheses. Experimental studies are one type of comparative studies [230] where two or more things are compared to discover their differences or similarities [231]. Shadish et al. [232] identify four types in their description of modern experiments, which include *Randomized Experiment*, *Quasi-Experiment*, *Natural Experiment*, and *Nonexperimental Designs*. In essence, an experiment

has independent x and dependent y variables. Through the planning and execution of an experiment, we investigate their relationship $y = f(x)$. The independent and dependent variables are known as cause and effect, respectively, when the goal is to identify a casual relationship. Confounding variables are extra variables that may have an impact on the perceived causal relationship but are not taken into account [231].

3.2.3 Experimental Design

A true experimental design compares two (or more) groups or situations where the cases to be tested are selected randomly [231, 233, 234]. A *quasi-experimental design* is similar to a true experiment design, except its randomness is not practical or feasible for a variety of reasons [231]. Campbell et al. [235] explain in their seminal work on experimental and quasi-experimental design that in a quasi-experimental design, the researcher lacks full control over the scheduling of experimental stimuli and the ability to randomize exposures. While the pretest-posttest control group design is widely used, the pretest concept is not actually essential to true experimental designs [235]. The posttest-only control group design controls most major threats to internal validity and is a more natural design than the pretest-posttest control group design [235].

In order to explore the differences between sMBSAP and Scrum development, here we use an experimental research strategy to probe this comparison. Specifically, we will describe the quasi-experimental posttest-only with nonequivalent groups study conducted. The nonequivalent groups' design is used when the control group and the experimental group do not have pre-experimental sampling equivalence [235, 236]. Rather, the groups constitute naturally assembled collectives.

First, all user stories ($m = 260$) were created and added to a *Product Backlog* (Step 1 in Figure 3.1). Bigger user stories were broken down into tasks (up to 20 tasks per user story). The user stories ($m = 260$) were reviewed and analyzed to deliver the core system modules (groups of related functions) through sprints ($n = 20$) (Step 2 in Figure 3.1). In order to create randomly assigned samples using intact clusters of sprints, all sprints ($n = 20$) identified in Step 2 were placed into one of two heterogeneous groups (Step 3 in Figure 3.1). These two groups, containing

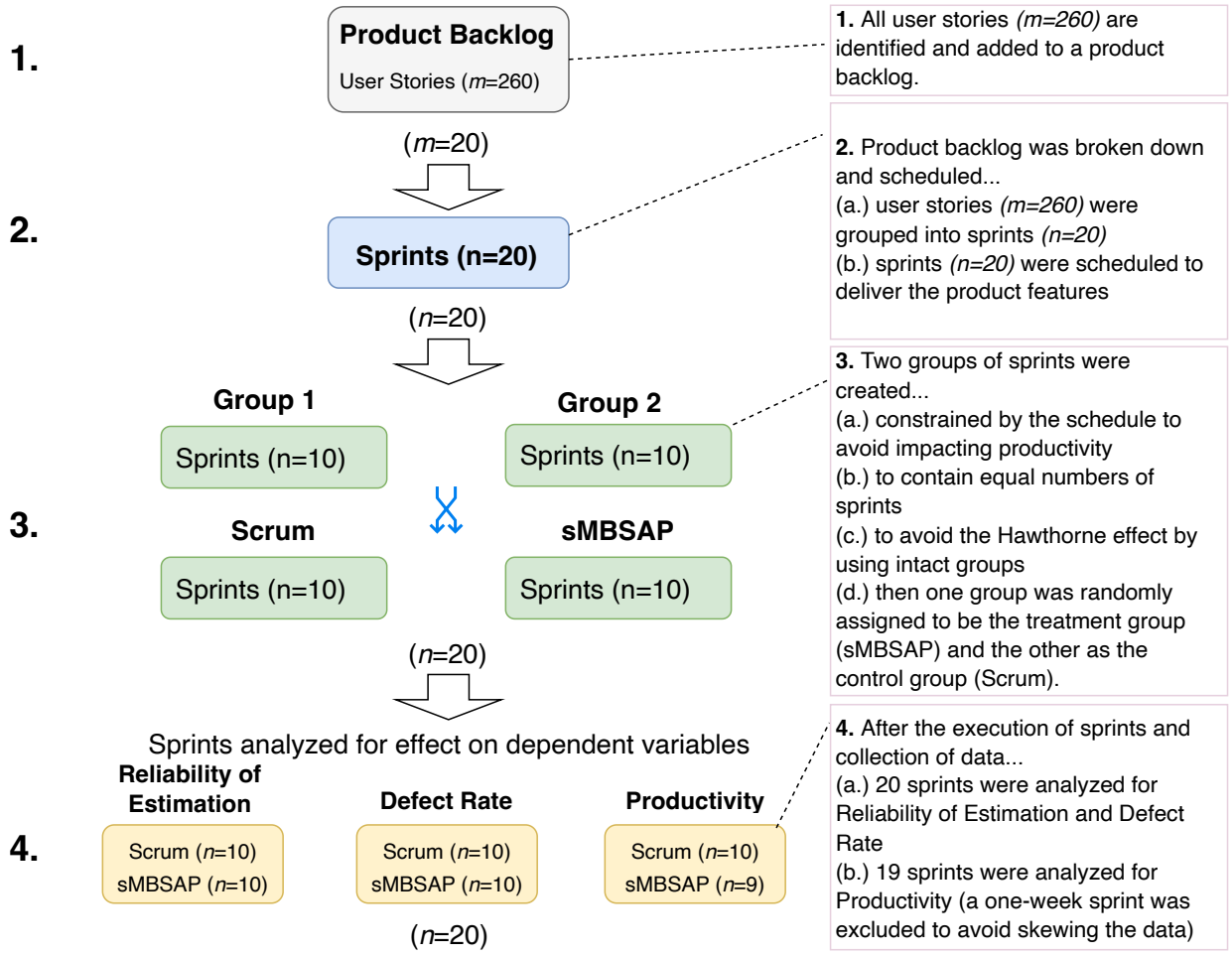


Figure 3.1: Random assignment of intact groups and subsequent collection of data.

equal numbers of sprints, were then randomly assigned, one to the treatment group ($n = 10$) sMBSAP and the other to the control group ($n = 10$) Scrum. These groups were formed in order to avoid the Hawthorn effect [237]. Moreover, the researcher created a model that includes a set of viewpoints and perspectives used for both approaches. This artifacts model was created to control one of the confounding variables, that is, "consistency of input artifacts", by ensuring as many similarities between the sprints as possible.

The strongest research design for experiments is true experimental design with random sampling from a given population to ensure equivalent groups [235]. Another slightly more elaborate nonequivalent group design called the pretest-posttest nonequivalent group design is used to assess the nature of initial selection differences between the groups and to take account of the effects of

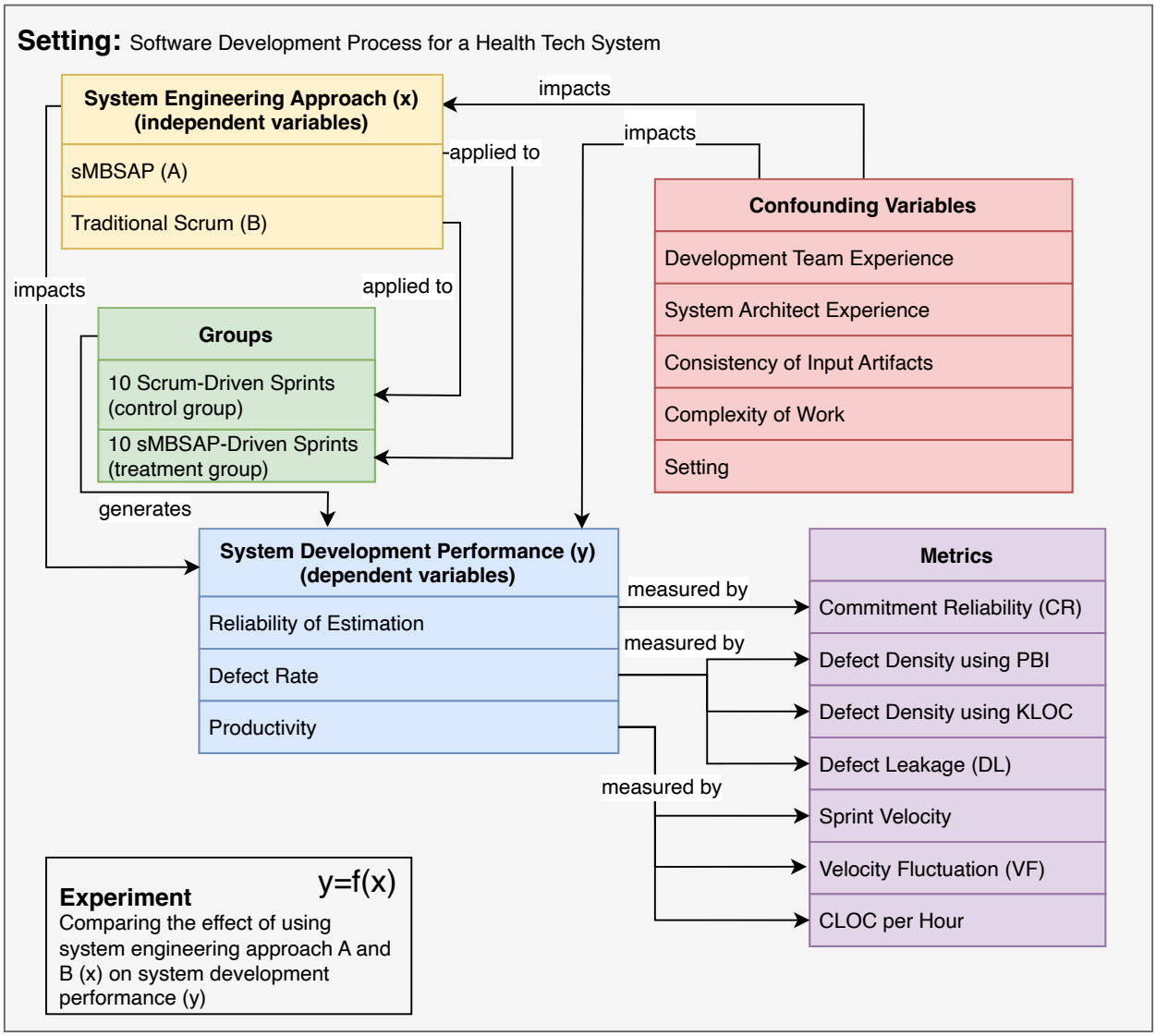


Figure 3.2: Experiment overview.

these selection differences [236]. However, within the context of this experiment, achieving random assignments without interrupting project delivery and momentum was found to be extremely difficult. The solution researchers offer to address this constraint is assigning the treatment to one intact group or the other randomly [235, 236]. The four steps used in the random assignment of the intact groups for this study are presented in Figure 3.1.

Now, the dependent variables are the software system development performance objectives, namely (1) *Reliability of Estimation* as measured by CR; (2) *Defect Rate* as measured by DD using *Product Backlog Items (PBIs)*, DD using $Thousands(Kilo)of Linesof Code(KLOC)$ and DL;

Table 3.1: *Product Development Team* personas.

Role	Gender	Industry Experience	Highest Education Level	Familiarity with System Development
System Architect and Scrum Master	Male	20+	MSc	Yes
Backend Developer	Male	15+	PhD	Yes
Frontend Developer	Male	15+	PhD	Yes
Health and Nutrition Scientist	Female	15+	PhD	No

(3) *Productivity* as measured by $SprintVelocity(SV)$, $VelocityFluctuation(VF)$, and *Count of Lines of Code* ($CLOC\ per\ hour$). The experiment overview illustrating the independent, dependent, and confounding variables is shown in Figure 3.2. This research is a longitudinal study where repeated observations of the same variables were collected for one year (between May 2020 and May 2021). The dependent variables from the executed sprints in both groups ($n = 20$) were collected and analyzed. Twenty sprints were executed and analyzed for *Reliability of Estimation* and *Defect Rate*. All executed sprints were two-week sprints except the last sprint, which was a one-week sprint. To avoid skewing the *Productivity* data, the one-week sprint was excluded from the analysis, so only 19/20 sprints were analyzed (Step 4 in Figure 3.1).

3.2.4 Experimental Setting

Context: The product development experiment took place within an early-stage, health technology startup company for one year (between May 2020 and May 2021), which aims to develop a health tech system that provides dietary recommendations to users based on their health profiles. The development period coincided with the COVID-19 pandemic when stay-at-home orders were in effect, so the *Product Development Team* primarily used virtual conferencing tools and Slack [238] for communication and collaboration.

Participants: The three co-founders, besides the lead author, served as the *Product Development Team*. The role of the lead author then was limited to developing the architectural artifacts based on requests from the team. Table 3.1 summarizes the *Product Development Team* personas.

It is often not feasible for any researcher to conduct an experiment that would involve developing the same system twice, either synchronously or asynchronously, using two different methodologies and two different teams because of the resources and time limitations. Therefore, the most practical method is to divide the development effort carried out by the same development team between the two approaches under comparison (independent variables), collect the data, and finally observe the effect of both approaches on the dependent variables.

3.2.5 Time Horizon

Saunders et al. differentiate between two types of studies from the time horizon perspective, *cross-sectional studies* and *longitudinal studies*. The research that focuses on a certain phenomenon (or phenomena) at a specific time "snapshot" is cross-sectional. The research that collects a diary or a series of snapshots as a representation of events over a given period is a Longitudinal study. Researchers seem to agree that longitudinal studies do not have to be for several years. A longitudinal study involves repeated observations of the same variables over short or long periods of time (i.e., uses longitudinal data) [223]. This research is a longitudinal study where repeated observations of the same variables (*Reliability of Estimation* data, *Defect Rate* data, and *Productivity* data) were collected for one year (between May 2020 and May 2021).

3.3 Techniques and Procedures

This section provides an overview of the five main activities carried out during the quasi-experiment and depicted in Figure 3.3, except for activity 3, which is described in detail in Chapter 2.

3.3.1 Step 1: Plan Experiment

The purpose of the first step is to control one of the confounding variables (shown in Figure 3.2), which is the consistency of input artifacts. At a high level, these artifacts are the information the *Product Development Team* receives from the *System Architect* at the beginning of a

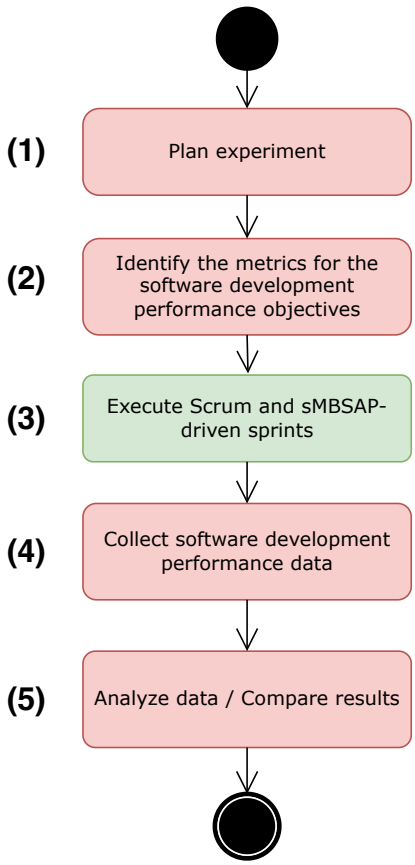


Figure 3.3: Activity Diagrams (AD) that shows the main experiment activities

sprint to use in building the software and writing the software code. An architecture framework defining the artifacts for each perspective and viewpoint of the system was developed [2]. The framework reduces the threat to validity by ensuring that both control and treatment groups are as similar as possible by ensuring that the *Product Development Team* receives comparable artifacts used in writing the software code.

This first step includes 3 substeps; Step 1.1 defines a unified framework for describing the system; Step 1.2 uses the unified framework to describe the system using the Scrum approach; and Step 1.3 uses the unified framework to describe the system using the sMBSAP approach.

Step 1.1: Define a Unified System Description (Architecture) Framework

An architecture framework was developed to capture the architecture content, guide the architecting and development processes, and provide views of the system from different perspec-

Table 3.2: The architecture framework used for developing the health technology system architecture and guiding the development process.

		Perspective			
		Structure	**Data**	**Behavior**	**Capabilities**
Viewpoint	**Operational**	**Classifications:** Hierarchy Levels **Product Breakdown:** Product Breakdown Structure, Structural Decompositions **Class Diagrams:** Domain Diagram with Specifications **Object Diagrams:** Decomposition of Domains to the level of Classes	**Classifications:** Hierarchy Levels **Conceptual Data Model (CDM):** Primary Data Entities, Foundation Classes **Logical Data Model (LDM):** Modeling Foundation Classes, Attributes and Operations **Physical Data Model (PDM):** Database Structure	**Use Case (UC) Diagram:** with Specifications **Activity Diagrams (AD):** with multiple Domains and Actors **Sequence Diagrams (SD):** Modeling non-stateful behavior of a group of classes	**Classifications:** Hierarchy Levels, User Story Types **Requirements:** Requirements Hierarchy, Requirement IDs, Requirements Texts, Verification Methods, Parents, Requirements Allocations
	Functional				
	Physical				

tives [205] (see Table 3.2). The architecture framework summarizes the system perspectives, viewpoints, and artifacts that guided the development process for Scrum and sMBSAP. This task also included identifying system *Domains* associated with the health tech system to help identify user stories, *Use Cases*, *Activity Diagrams (AD)*, Sequence Diagrams (SD), and mock designs of the system screens. The framework was synthesized from the framework defined in the MBSAP methodology [1].

A table format was adopted due to its visual representation, which aids in communicating, understanding, and tracking the architecting process [205]. The structure, data, *Behavior*, and *Capabilities* perspectives from MBSAP were chosen for the table columns. The *Operational*, *Functional*, and *Physical Viewpoint (PV)* from MBSAP were chosen for the rows of the table, which also align with the axis of abstraction from the system architecture taxonomy (see Figure 1.4).

Step 1.2: Synthesis of the description for the Scrum-driven sprint

A description of the Scrum process is explained in Chapter 2. The process shows the four main meetings that occur during the Scrum cycle: *Sprint Planning*, *Daily Standups*, sprint review, and *Sprint Retro* [24, 95, 160, 161]. Scrum artifacts are the key information the development team uses to execute the product development through Scrum-driven sprints and show the progress to

Table 3.3: Information model for the Scrum approach showing the typical artifacts generated in each perspective.

		Perspective			
		Structure	**Data**	**Behavior**	**Capabilities**
Viewpoint	**Operational**	Product Breakdown: Initiatives, Epics, and Stories	Data Model Slides	User Stories Flowcharts	User Stories (US), Product Backlog, Sprint Backlog
	Functional				
	Physical				

stakeholders. The main Scrum artifacts include *Product Backlog*, *Sprint Backlog*, and Increments [24, 95, 160, 161].

A high-level model of the Scrum artifacts was developed from the architecture framework in Table 3.2, covering the system description artifacts for each perspective and viewpoint (see Table 3.3). The Scrum artifacts are developed and maintained using Agile and document-based tools. ClickUp [163], an Agile project management tool, was used to capture *PBIs*, which are organized in *Product Backlogs* and *Sprint Backlogs*. Microsoft Tools such as PowerPoint, Word, and Excel were utilized to capture and communicate numerical, textual, and graphical system information or artifacts, such as *Product Breakdown*, "Conceptual Data Model", technology stack, and other information.

Agile practitioners devalue comprehensive documentation; therefore, the *Use Case Diagrams* and *Data Models* are not typical in Scrum methodology [25, 159]. *User Stories* are used to capture scenarios or system *Behavior* by describing how the user interacts with the system [25]. Younse et al. encapsulated model diagrams in a tabular format to present synthesized information [205]. A similar synthesis was used to present the complete set of artifacts developed for the Scrum-driven sprints shown in Figure 3.4 and organized within the architecture framework developed in Table 3.2. The definition of the key Scrum terms and artifacts can be found in Chapter 2.

During each step of the Scrum process (Figure 2.1 in Chapter 2), information was created and then added to the appropriate artifacts specified in Figure 3.4. With the Scrum approach, informa-

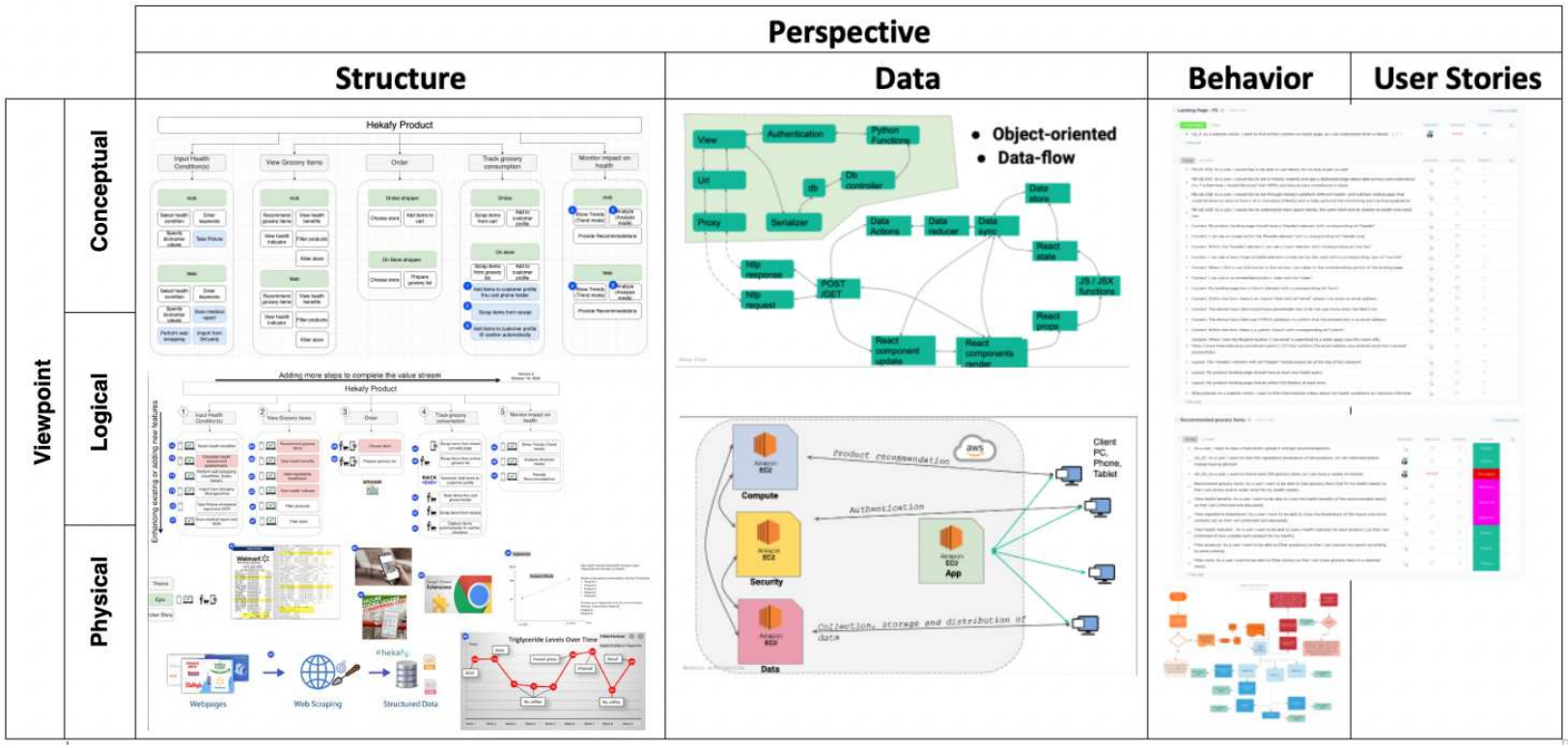


Figure 3.4: The complete set of artifacts developed for the health technology system using the Scrum methodology.

tion elements were captured in an Agile tool, ClickUp [163], and amongst separate, independent documents (rather than a primary, integrated system model as in the MBSAP approach).

Step 1.3: Synthesis of the description for the sMBSAP-driven sprints

Similar to Scrum, a description of the sMBSAP artifacts for the health technology system was synthesized using an MBSE tool within the architecture framework developed in Table 3.2 in Step 1.1. The sMBSAP synthesized artifacts are displayed in Figure 2.9. A description of the sMBSAP process is explained in further detail in Chapter 2.

3.3.2 Step 2: Identify the Metrics for the Performance of Software Development

Metrics for Reliability of Estimation

The primary metric used to assess the *Reliability of Estimation* is *Commitment Reliability* CR for each completed sprint:

$$CR = \frac{Completed\ SP}{Planned\ SP} \tag{3.1}$$

In order to assess the CR for all Scrum-driven sprints vs. all sMBSAP-driven sprints, the average CR was used. The average CR using a particular approach is:

$$Average\ CR = \frac{\sum CR}{Number\ of\ Sprints} \tag{3.2}$$

Metrics for Productivity

The primary metric used to assess *Productivity* is SV, that is the sum of capabilities (features, *User Stories*, *Requirements*, or *PBIs*) successfully delivered in a sprint, measured in $StoryPoints(SP)$, as:

$$SV = \sum Completed\ SP\ During\ a\ Sprint \tag{3.3}$$

In order to assess the SV for all Scrum-driven sprints vs. sMBSAP-driven sprints, the average SV was used. The average SV using a particular approach is:

$$Average\ Velocity\ = \frac{\sum Sprint\ Velocity}{Number\ of\ Sprints} \tag{3.4}$$

VF is another way to monitor SV, and it represents the SV variance from average. Monitoring VF helps analyze the root cause of the team missing their commitment [109] and is calculated with:

$$VF = \frac{Average\ Velocity - Sprint\ Velocity}{Average\ Velocity} \tag{3.5}$$

Finally, the total $CLOC\ per\ hour$ was used as a secondary metric to assess *Productivity*:

$$CLOC\ per\ hour = \frac{CLOC}{Development\ Duration\ (hrs)} \tag{3.6}$$

Metrics for Defect Rate

The primary metric used to assess the *Defect Rate* is DD which is the number of defects found within PBIs executed during a sprint as:

$$DD = \frac{Defect\ Counts \text{ (pre-delivery and post-delivery)}}{Size \text{ (measured in PBIs)}} \tag{3.7}$$

where "pre-delivery" defects refer to the defects found during the testing phase, while "post-delivery" defects refer to the defects found after the testing phase.

In order to assess the DD for all Scrum-driven sprints vs. sMBSAP-driven sprints, the average DD was used. The average DD using a particular approach is:

$$Average\ DD = \frac{\sum DD}{Number\ of\ Sprints} \tag{3.8}$$

Another way to calculate the DD is by assessing the number of defects found within KLOC. DD using KLOC using a particular approach is calculated as:

$$DD = \frac{Defect\ Counts \text{ (pre-delivery and post-delivery)}}{Size \text{ (measured in KLOC)}} \tag{3.9}$$

Finally, the DL was used as a secondary metric to assess *Defect Rate*. It represents the ratio of defects found during the next phase (post-delivery) vs. defects found during testing (pre-delivery) measured in defects count. Monitoring DL would help analyze the root cause of leaked defects in order to avoid them before leakage [112] and is calculated with:

$$DL = \frac{Defect\ Count \text{ (post-delivery)}}{Defect\ Count \text{ (pre-delivery)}} \tag{3.10}$$

In order to assess the DL for all Scrum-driven sprints vs. sMBSAP-driven sprints, the average DL was used. The average DL using a particular approach is:

$$Average\ DL = \frac{\sum DL}{Number\ of\ Sprints} \tag{3.11}$$

3.3.3 Step 3: Execute Scrum and sMBSAP Drive-Sprints

The processes for executing the Scrum method and sMBSAP methods are explained in Chapter 2.

3.3.4 Step 4: Collect System Development Performance Data

Planning Poker [66] and Fibonacci scale [239] techniques were used for planning the amount of work for all sprints. Planning Poker is a Scrum estimation technique that determines relative sizing using $StoryPoints(SP)$ and playing cards [67]. During *Sprint Planning*, the *PBIs* are captured in an Agile tool such as ClickUp [163]. During sprint execution, the tracked data elements for each PBI are updated daily. The updated data include the start date, due date, actual closing date, planned $StoryPoints(SP)$, and $StoryPoints(SP)$ completed. At the end of each day, the

completed *PBIs* are marked as"closed", and the closing date is noted along with the sprint duration. Also, the $StoryPoints(SP)$ associated with *PBIs* were summed up.

The completed $StoryPoints(SP)$ are plotted on a burnup chart, allowing a visual comparison between ideal and actual progress. At the end of each sprint, the *Reliability of Estimation* and *Productivity* metrics were calculated. The data were extracted from ClickUp to an Excel Spreadsheet, then imported to PowerBi [240] for analysis and visualization.

To monitor defects, the tester logged the defect under the relevant list using a GitHub repository. Based on the nature of the defect, the tester assigned the defect to a member of the *Product Development Team*. The product manager reviewed the logged defects, prioritized them, and ensured that each one was assigned to the right person to address that defect. The testers were encouraged to provide as many details as possible when logging a defect on GitHub. They were encouraged to provide steps to reproduce the error and provide screenshots for the defect and annotate the screenshot for additional clarity. The person who was assigned to a defect may ask questions or add comments, and the tester may respond till the defect is marked as closed and verified by the product manager.

At the end of each sprint, the defects found during testing (pre-delivery) and after passing testing (post-delivery) were captured using GitHub. At the end of the last sprint, the defect-related data points were captured including sprint duration, PBIs, defects captured pre-delivery, defects captured post-delivery, and CLOC.

3.3.5 Step 5: Analyze Data and Compare Results

Descriptive statistics, including means and standard deviations, were used to describe and summarize data to discover emerging patterns. Descriptive data were also used to examine the dispersion of the data sets concerning their mean and standard deviation and visually inspected using boxplots and bar charts.

The normality test was also used for all three dependent variables [241] to assess the normal distribution. The unpaired t-test was also conducted. The unpaired t-test is a type of inferential

statistic that shows how significant the differences between two groups are. In other words, it shows whether those differences (measured in means) could have happened by chance. The t-test is considered a robust test against the normality assumption [242, 243]. The assumptions required for the independent samples t-test were evaluated prior to conducting each t-test and are discussed in Section 3.4.4. The descriptive statistics, normality test, and t-test were performed using GraphPad Prism version 8.0.0 for Mac [244].

3.4 Quality of the Research Design

3.4.1 Reliability and Validity

In the context of the discussion about the quality of research and its findings, Raimond [245] proposed the 'how do I know?' test, that is, 'Will the evidence and my conclusions stand up to the closest scrutiny?'. The answer is that you cannot know if you take the question literally. You can only lower the likelihood that you will provide the incorrect result. This justifies how critical solid research design is [245]. Rogers [246] summarized this by clarifying that scientific research is a way of preventing the researcher from deceiving themselves in regard to their creatively formed subjective idea, which has developed out of the relationship between the researcher and their material.

Saunders et al. [216] define validity as "the appropriateness of the measures used, accuracy of the analysis of the results, and generalisability of the findings". Campbell et al. [235] explained two types of validity: internal validity and external validity, and the factors that jeopardize them. Reliability, on the other hand, refers to replication and consistency [216]. In other words, a research would be considered reliable if it can be successfully replicated and the same results are achieved. Internal validity refers to the extent the findings can be attributed to the intervention under investigation rather than to flaws in the research design [216]. External validity, on the other hand, is concerned with the possibility of generalizing the research findings to other relevant contexts.

3.4.2 Minimizing Threats to Reliability

The threats to reliability include an error or bias from the participant's or researcher's side. Although the *Product Development Team* is not the subject of the experiment, the researcher attempted to minimize any error or bias from their side. The researcher identified four sources that could potentially pose threats to reliability: (1) participant error, (2) participant bias, (3) researcher error, and (4) researcher bias. In order to mitigate the first threat, the researcher ensured control of any factor which could adversely alter how a member of the *Product Development Team* behaves. As for the second threat, the researcher has not observed any inconsistency in how the *Product Development Team* planned or executed the sprints. Also, the researcher reviewed all defects detected internally and externally, and no known inconsistencies have been observed.

To mitigate the third threat, the researcher did not hesitate to ask the *Product Development Team* for clarification when the researcher was in doubt during any phase of the experiment. A researcher's bias may come from any factor that induces bias in the researcher's input recording [216]. As indicated earlier, given the nature of this quantitative experiment, the interpretation is very limited, and the reliance is mainly on the collected quantitative data. The possibility of the researcher's bias when recording the data, as they are generated by the *Product Development Team*, is very limited. For example, when the team reports 20 defects during a sprint, the researcher would record this number without further interpretation from his side.

3.4.3 Minimizing Threats to Validity

A variety of factors may jeopardize experimental and quasi-experimental research's internal validity. The threats are listed below, along with an explanation of how the researcher sought to reduce them.

1. Selection bias: In this study, some selection bias was introduced by the fact that sprints were chosen to be in one group or the other based on the scheduling of the sprints. The *Product Development Team* thought it would be counterproductive to execute one sprint using one approach and the following related sprints with the other approach. Although

this non-random assignment is considered a threat to internal validity, the benefit of it is that it mitigated the risk of impacting the *Product Development Team* momentum, which, if occurred, would have affected the *Productivity* measures. Further details on the steps followed to apply random assignment of the intact groups are provided in Section 3.2.3.

2. Testing effect: Two measures have been taken to address this threat. First, having a control group that was executed with Scrum helped guard against this threat since the sprints of the control group were equally subject to the same effect. Second, the researcher did not share with the *Product Development Team* the details of the study or the specific variables under investigation.

3. History effect: has not been observed or occurred during the experiment.

4. Instrumentation effect: has not been observed or occurred during the experiment.

5. Mortality effect: has not been observed or occurred during the experiment.

Experimental and quasi-experimental research's external validity may be also jeopardized by a variety of factors. The factors jeopardizing external validity are listed below, along with the mitigation strategies which the researcher implemented.

1. Changes of causal relationship due to variations of the implementation within the same approach. During the execution of both groups of sprints, no variations were observed.

2. Interaction of causal relationship with settings. This factor considers the setting in which the cause-effect relationship is measured, jeopardizing external validity [223]. The researcher believes that the experimental setting is similar to most development projects after COVID-19, in which *Team Members* work and collaborate remotely. However, the research acknowledges that conducting this experiment in other settings would provide further insights into this factor.

3. Interaction of causal relationship with outcomes. This outcome refers to the fact that a cause-effect relationship may exist for one outcome (e.g., more accurate *CLOC*) but not another seemingly related outcome (e.g., *Productivity*) [247]. The researcher studied the impact of the two approaches on three outcomes. The established causal relation has not been extended from one outcome to another without data collection and measurement, which gives a fuller picture of the treatment's total impact.

4. The reactive or interactive effect of testing [235]. Given that this quasi-experiment is a posttest only, this factor does not apply to this experiment.

3.4.4 Minimizing Threats to Statistical Conclusion Validity

Threats to statistical conclusion validity are about answering the question, "did the investigators reach a correct conclusion about whether a relationship between the variables exists in the population or about the extent of the relationship?" [234]. Certain assumptions were assessed using various statistical tests before conducting quantitative tests.

Examining the dispersion of the data and screening for errors prior to conducting an in-depth analysis is important [248]. All system development performance data were screened for missing items or inconsistencies. To accurately and reliably interpret test results, some assumptions essential to all data analyses should be proven to be maintained [249]. The assumptions and related tests used before performing the independent samples t-test are considered and discussed in this section. Independent t-tests involve the following assumptions: (1) random assignment, (2) independence, (3) level of measurement, (4) normality and outliers, and (5) homogeneity of variance.

1. Given that the two groups of sprints are not assembled randomly, due to scheduling constraints, the two groups are considered non-equivalent (preexisting groups). The schedule has been found to play a role in the assignment of subjects to experimental groups in other quasi-experiments in software engineering [250]. In such cases, researchers suggest assigning the treatment to one of the preexisting (intact) groups or the other randomly [235, 236]. This approach has been borrowed and implemented by researchers in various fields [251],

including software engineering [250] and has also been implemented in this experiment. Therefore, the assumption of random assignment of intact group was tenable.

2. The assumption of independence suggests that data are gathered from groups that are independent of one another [242,252]. In this experiment, special consideration has been given to the sequence of the sprint implementation to better ensure that the independence assumption holds for the observations within each group. The independence assumption for the t-test refers to the independence of the individual observations within each group rather than the interdependencies between sprints. To best satisfy this assumption, each observation should be unconditionally unrelated and independent of the others in terms of data collection or measurement. Violations of the independence assumption, such as having repeated measures or correlated observations, can lead to biased or incorrect results when using the independent t-test. As an example from the context of this experiment, the number of defects observed in Sprint 6 (a Scrum-driven sprint) is independent of that observed in Sprint 7 (a Scrum-driven sprint), and is also independent of that observed in Sprint 8 (an sMBSAP-driven sprint). A higher or lower number of defects in Sprint 6 has no relationship to the number of defects in Sprints 7 or 8, although there is a schedule interdependency between the three sprints. In summary, while the sprints may have scheduling dependencies, given that the observations within each group are generally independent of each other, the assumption of independence was tenable for the independent t-test.

3. The dependent variables must be continuous and measured at the interval or ratio scale in order to satisfy the level of measurement assumption for the independent t-test. The level of measurement assumption also requires a categorical independent variable with two groups, one treatment and one control [242]. The type of data collected (ratio scale and interval data) and having two groups satisfy this assumption.

4. The assumption of normality for each set of system performance data, where the mean was calculated, was visually evaluated; first, using a boxplot to eliminate outliers; then, using a bar chart; and then statistically calculated using the normality test [241].

5. The Levene's test is used to assess the argument that there is no difference in the variance of data between groups. A statistically significant value (0.05) denotes that the assumption has not been satisfied and that the variance between groups is significantly different. Equal variance was not assumed when the Levene's test was significant. Similarly, an equal variance was assumed when the Levene's test was not significant ($p > 0.05$) [253].

3.5 Research Methodology Summary

This chapter introduces the research methodology. There are four stages that have been undertaken to conduct this research project. Each stage is concerned with identifying one layer of the research methodology. These layers include philosophy, approach to theory development, research design, and techniques and procedures. The research philosophy and approach are briefly explained.

The research design, which includes (1) methodological choice, (2) research strategy, and (3) time horizon, are explained. The methodological choice of this research falls under the quantitative methods category. This study used an experimental research strategy to address the overarching research question and hypotheses, specifically a quasi-experimental posttest-only with nonequivalent groups. This research is a longitudinal study where repeated observations of the same variables (*Reliability of Estimation* data, *Defect Rate* data, and *Productivity* data) were collected for one year (between May 2020 and May 2021).

The techniques and procedures for the study and their associated activities are explained. Finally, the steps taken by the researcher to mitigate threats to reliability, validity, and validity of statistical analysis are discussed.

Chapter 4

Results and Discussion[4]

The purpose of this chapter is to discuss the implementation of the Scrum Model Based System Architecture Process (sMBSAP) approach and presents the results of the data analysis for the experiment. The chapter is structured into three main sections; the first section (Section 4.1) discusses the implementation of the sMBSAP approach and provides a side-by-side comparison for Scrum, Model-Based System Architecture Process (MBSAP), and sMBSAP to aid the reader in understanding the similarities and differences among the three approaches. The section also explains the key characteristics and benefits of implementing the sMBSAP approach. The second section (Section 4.2) is dedicated to reporting the statistical analysis, including the independent t-test for each dependent variable, including *Reliability of Estimation* results (4.2.1), productivity results (4.2.2), and *Defect Rate* results (4.2.3). Finally, this section provides additional observations (4.2.4) captured during the experiment. The chapter concludes with the last section (Section 4.3), which summarizes the results and discussion.

4.1 Discussion for sMBSAP

In addition to the main characteristics of Scrum and the Model-Based System Architecture Process (MBSAP), the sMBSAP is also concerned with the engagement of the *Product Development Team* in customizing the MBSE tool, using Unified Modeling Language (UML)-based and non-UML-based models to describe the system, and leveraging the built-in models (provided in some tools) to get to an initial version of the model more quickly. Moreover, the Scrum ceremonies are mapped and integrated into the entire sMBSAP lifecycle with some modifications. The sMBSAP has the same cyclic shape as Scrum to demonstrate that the development process is iterative. The iterative nature applies to both the product delivery and the system model construction.

[4]Parts of this chapter previously appeared in [2, 3].

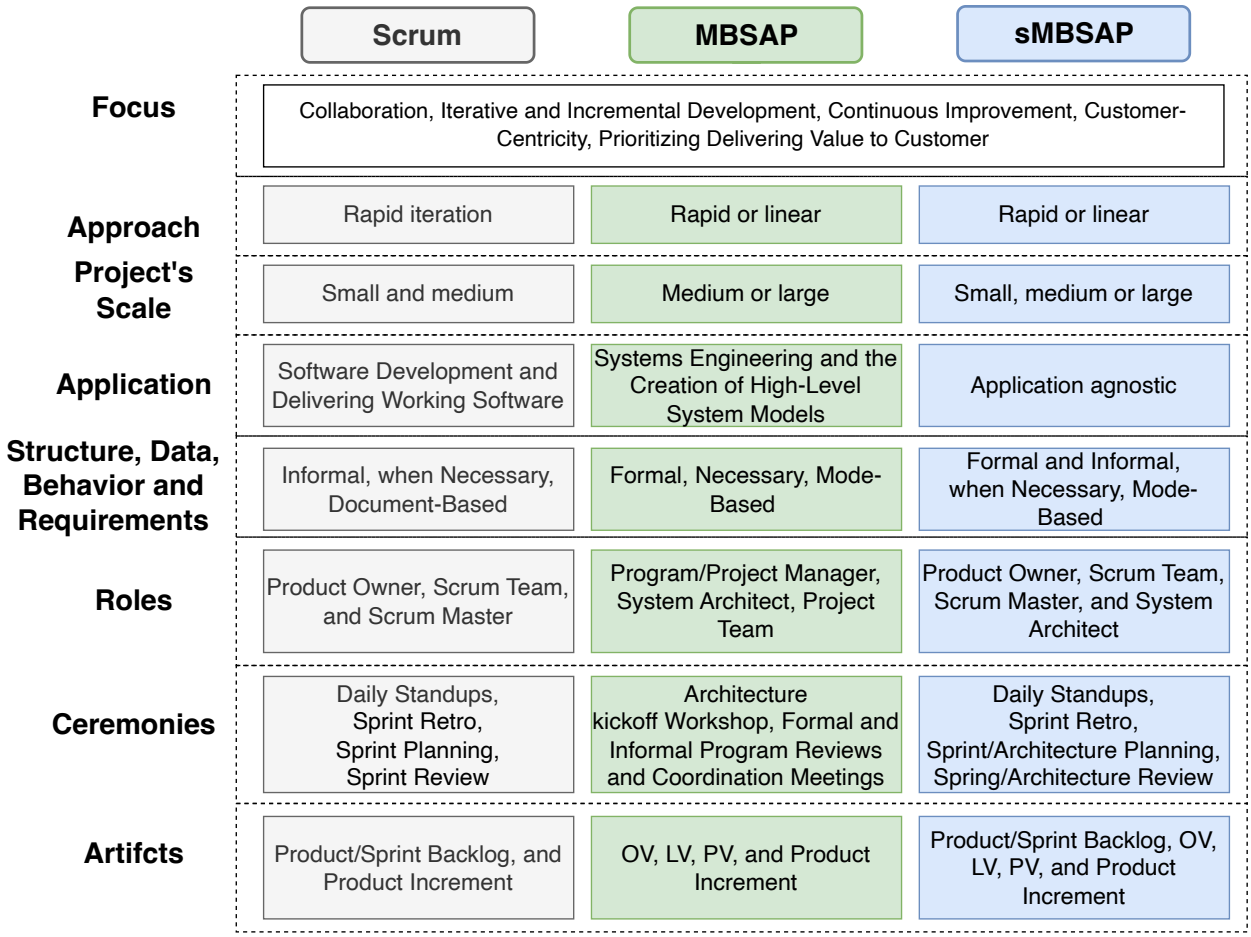


Figure 4.1: Comparison of Scrum, MBSAP, and sMBSAP.

Figure 4.1 shows a comparison among Scrum, MBSAP, and sMBSAP. The comparison reveals that Scrum and MBSAP have similarities, including a focus on collaboration, iterative and incremental development, and continuous improvement. In addition, both approaches value customer-centricity, prioritizing delivering a working product to the customer, and responding to change. These similarities have been passed down to the sMBSAP. However, there are also differences among Scrum, MBSAP, and sMBSAP. Scrum prioritizes working software as the primary measure of progress over system documentation. Scrum documentation does not follow a formal modeling language. While the MBSAP values a working product, it places a great emphasis on using models to capture and communicate system information. The sMBSAP, on the other hand, is a middle point, as it uses both formal and informal modeling languages to keep system information within the model. The model can be customized to keep the details of system architecture at a high

level or comprehensive. Additionally, Scrum is primarily focused on software development. However, the MBSAP is more focused on systems engineering and the creation of high-level system models. The sMBSAP, on the other hand, is application agnostic and can be applied to software, defense, or other industries. As for project size, Scrum is primarily used for small to medium-sized projects. However, the MBSAP is more geared towards medium to large-sized projects. Finally, the sMBSAP can be customized to fit small to large projects.

Unlike traditional document-based methods, an MBSE tool is the key to handling, processing, and executing the data and information generated or collected during the system development process. Therefore, it is important to select the appropriate tool to create a modeling environment that fits the different kinds of data and information being processed. A proper MBSE tool can simplify the working process and accelerate the working efficiency. The MBSE tool used for architecting the health tech system in this pilot study was Sparx EA [186]. Sparx EA was selected due to its compatibility and readiness for software development models.

It is important to note, however, that in an MBSE-driven environment, having the best tools in the wrong environment would not contribute to project success. What contributes to project success is having the right group of individuals in product development. Even more crucial is how these people interact with one another. The other factor contributing to success is building a feedback loop with the customer to ensure that successful *Product Increments* are delivered. This feedback loop will open the door to embracing change, which always happens. These factors are inherited from both Scrum and MBSAP for sMBSAP, and they align well with the four values of the Agile Manifesto [54].

When a change is requested by the customer in the middle of a sprint, it is suggested to add the created *User Story* to the *Product Backlog* and reprioritize the *Product Backlog Items (PBIs)* rather than adding the *User Story* to the current *Sprint Backlog*. The benefit of this way of handling change is that it would avoid assuming that the *Development Team* would finish their work in progress and would be able to begin and finish the added *User Story* by the end of the sprint. The

more assumptions a project has, the more risk exposure it has. Moreover, adding *User Stories* to an ongoing sprint would impact the monitoring of Estimation Reliability and Velocity.

During the execution of the phases of the sMBSAP approach, data were generated or collected from the beginning to the end of the health tech system development effort. Keeping the data and artifacts in one model made accessing and retrieving data easier compared to the process when using document-based methods. Tracking back the *Requirements* (*User Stories*) or even performing simple simulation tasks for validation and verification was also beneficial. The key characteristics and benefits of implementing the sMBSAP include the following:

- The *System Architect* works closely, not in a silo, with the *Product Development Team* to (1) co-customize the MBSE tool at the beginning of the project to align with the needs of the project and the *Product Development Team*. The customization exercise is used as an MBSE infusion opportunity. In an MBSE-driven project, the *Product Development Team* is the first customer of the system architecture, and the Maintenance and Operations team is the end user, as they leverage the system architecture in operating software applications, monitoring system performance, making defect repairs, etc. The *System Architect* would need to work closely with different business and technical stakeholders from the customer organization to ensure that the model perspective and viewpoints are customized to fit their needs and the organization's standards.

  (2) It is also necessary to engage and educate the *Product Development Team* about the basic concepts and processes of the architecture and (3) to empower and support the *Product Development Team Members* (owners of core components of the system) to define discipline-specific MBSE methods. For example, a frontend developer develops a wireframe for the "Health report summary" including the screen elements, such as the following: buttons—"View my groceries"; dashboard indicators and messages—"Health score" and "Summary health report"; navigation sections—"Dietary", "Lifestyle", "Disease risk", "Organ health", and "Mental health"; finally, a scroll bar. The *System Architect* works closely with the frontend developer to ensure that every screen element is aligned with and mapped to *Require-*

ments or *User Stories*, as shown in Figure 4.2. The *System Architect* adds the relevant *Requirements* and *User Stories* to the wireframe. Throughout the process, the wireframe is refined and updated to reflect the intended use of each screen. The outcome of this collaborative effort is a model-based wireframe (an artifact unique to the sMBSAP).

- Selecting an MBSE tool with (1) built-in methodologies that support the transformation of current systems engineering practices into model-centric engineering practices and (2) built-in models that support specific *Use Cases* would help create a faster first iteration of the model. This fast turnaround increases the engagement of the *Product Development Team* and contributes to changing the perception that architecture modeling slows down the development process. For example, some MBSE tools have built-in Gantt charts, which can automatically display the schedule for sprints, and built-in and customizable dashboards that can be used to show the progress of a sprint. Moreover, the *Product Owner*, *Scrum Master*, and *Team Member* roles can all be supported, needless to say, by the role of the *System Architect*. Selecting and customizing the right MBSE tool will provide a cohesive collaboration and*Requirement* management platform.

- The sMBSAP enables the *System Architect* to use Agile terminologies that the *Product Development Team* understands. Implementing Agile concepts such as sprints, *Product Backlog*, *Epics*, and *User Stories* conveys a sense of familiarity to the *Product Development Team*, even if these concepts are implemented within the context of an MBSE and architecture-driven environment.

- The sMBSAP leverages the MBSE tool to combine the UML-based formal description of the system with non-formal models that fit the needs of the *Product Development Team*. Combining formal and non-formal modeling aids in addressing the usability challenge, as it gives more freedom to the *Product Development Team*. The value of this combination is to instill in the *Product Development Team* the concept of keeping all artifacts in one model. For example, wireframes are valuable visuals that are widely used in Agile software

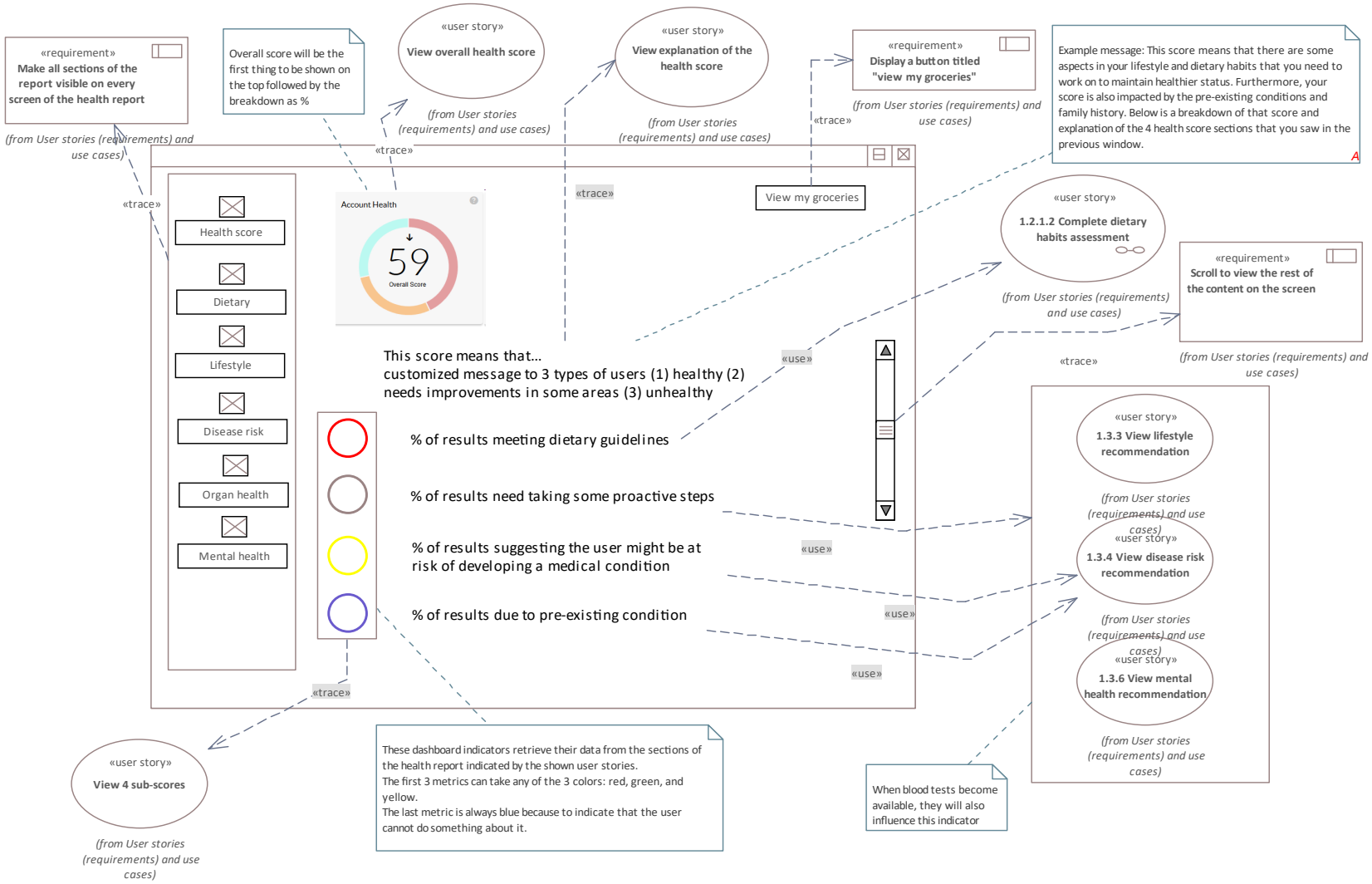


Figure 4.2: *Requirements* and *User Stories* traced to elements of a wireframe diagram embedded in the sMBSAP model.

development. Integrating the non-UML-based wireframes in the sMBSAP approach could increase the engagement of the *Product Development Team* and the adoption of the sMBSAP.

- The architecture effort progresses one sprint at a time. After the *System Architect* prepares the high-level end-to-end Operational Viewpoint (OV) of the system, they focus only on the *Requirements* and the *Structural*, *Behavioral*, and *Data* perspectives of one sprint at a time. In that sprint, the OV is further elaborated into an LV, which will be progressively elaborated in future sprints. This approach could potentially be a step toward addressing the disconnect between how the system is conceptualized and how it is described since the description is progressively elaborated over several weeks or months.

- The *Requirements* and *User Stories* will be traced to the various perspectives of the system model throughout the subsequent iterations or viewpoints. In *Use Case Diagrams*, *Requirements* are traced to *User Stories* that are modeled using a stereotyped Use Case. *Requirements* can then be shown on other models to trace the implementation of *Requirements* and *User Stories* (a *Requirement* or *User Story* is created once and used multiple times across the model). This approach simplifies the *traceability* process and bypasses the need to develop and maintain a document-based *Requirement Traceability Matrix*. Model-based *Requirement Traceability* could be a meeting ground between those who devalue *Traceability* and those who want to align with procurement/reporting practices.

- The iterative benefits of Agile combined with an integrated model of artifacts, as proposed in the sMBSAP, could be a practical happy medium between light documentation enthusiasts and those who value heavy documentation. The centralized management of artifacts makes the sMBSAP approach suitable for projects that value a working product and, at the same time, are keen to have more manageable, accessible, and retrievable documentation via a system architecture model.

On the other hand, there are a few challenges related to the adoption of the sMBSAP. Some of these challenges include the perception that MBSE is performed by a tool, although several

researchers explained that MBSE is more than just a tool [171, 183, 191]. Selecting an MBSE tool without a deep understanding of user needs is another challenge that may impede the adoption of the sMBSAP. Transitioning to a model-based software engineering approach requires a high level of executive support, which may not be always present. Finally and most importantly, adopting the sMBSAP requires a considerable investment in training because of its steep learning curve. Organizations may implement organizational change management initiatives to facilitate organizational adoption, but such organization-wide initiatives themselves require investment and management support.

4.2 Results and Discussion for the Quasi-Experiment

This section compares the *Reliability of Estimation*, *Productivity*, and *Defect Rate* metrics using Section 3.3.2 for both the Scrum-driven and sMBSAP-driven sprints. The comparison of *Reliability of Estimation* is conducted based on the results of the CR. The comparisons of *Productivity* are conducted based on the results of the SV, VF, and $CLOC\ per\ hour$. Finally, the comparisons of *Defect Rate* are conducted based on the results of the DD and DL. Abbreviations used in this section include the number of samples (N), arithmetic mean (M), and standard deviation (SD).

4.2.1 Reliability of Estimation Results

4.2.1.1 Commitment Reliability (CR) Comparison

The CR of each sprint in chronological order is presented in Figure 4.3. For the Scrum-driven sprints, the lowest CR was 0.71 for Sprint 2, while the highest CR was 0.86 for Sprint 12. For the sMBSAP-driven sprints, the lowest CR was 0.87 for Sprint 3, while the highest CR was 1.0 for Sprint 20. The Scrum-driven sprints ($N = 10$) were associated with a *Commitment Reliability* CR ratio of $M = 0.81$ ($SD = 0.046$). By comparison, the sMBSAP-driven sprints ($N = 10$) were associated with a numerically larger *CR* ratio of $M = 0.93$ ($SD = 0.032$). The descriptive statistics are shown in Table 4.1.

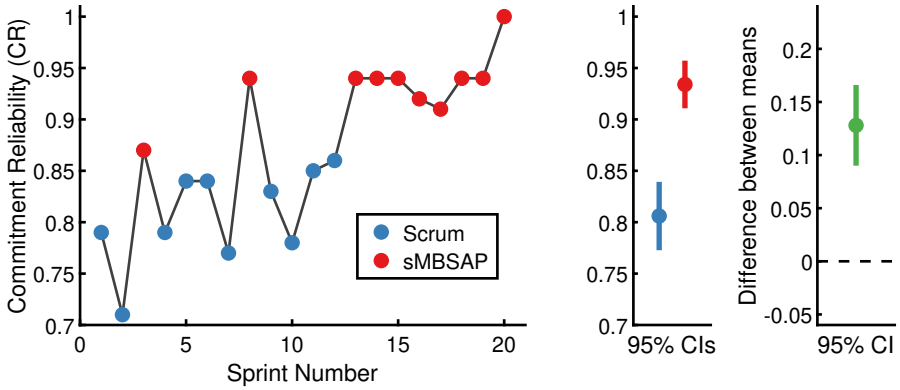


Figure 4.3: *Commitment Reliability (CR)* results.

Table 4.1: Descriptive statistics associated with *Commitment Reliability* (CR).

	N	M	SD	Skew	Kurtosis
Scrum	10	0.81	0.046	-0.861	0.45
sMBSAP	10	0.93	0.032	0.041	2.83

To test the hypothesis that the Scrum and sMBSAP were associated with a statistically significantly different mean *CR* ratio, an unpaired independent sample t-test was used to compare the Scrum-driven sprints and sMBSAP-driven sprints. An alpha level of 0.05 was utilized. The Scrum-driven sprints and sMBSAP-driven sprints distributions were sufficiently normal for the purpose of conducting the t-test (i.e., skew $< |2.0|$ and Kurtosis $< |9.0|$ [254]). Additionally, the assumption of homogeneity of variances was tested and satisfied via Levene's F test, resulting in $F(18) = 2.059$ and $p = 0.297$. The independent samples t-test was associated with a statistically significant effect: $t(18) = 7.15 \rightarrow p < 0.0001 \ll 0.05$. Thus, the sMBSAP-driven sprints were associated with a statistically significant larger mean *CR* than the Scrum-driven sprints. A graphical representation of the means and the 95% confidence intervals are displayed in Figure 4.3.

4.2.2 Productivity Results

4.2.2.1 Sprint Velocity (SV) Comparison

The SV of each Scrum-driven sprint in chronological order is presented in Figure 4.4. The lowest SV was 22 for Sprint 2, while the highest SV was 30 for Sprint 12. The Scrum-driven sprints ($N = 10$) were associated with a SV of $M = 26.8$ ($SD = 2.3$). Similarly, the SV of each

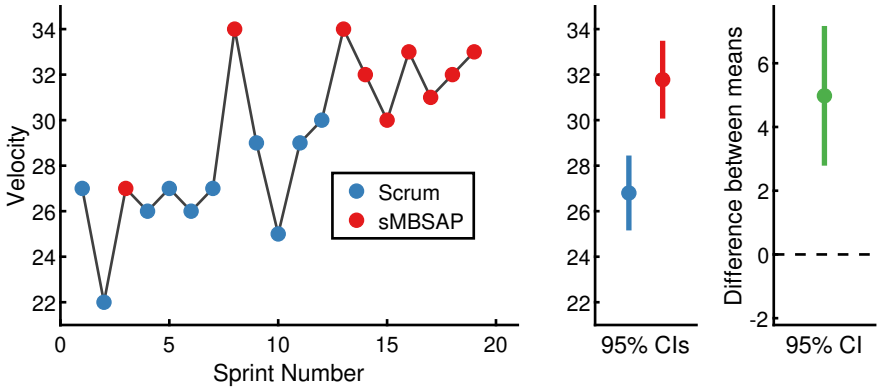


Figure 4.4: *Sprint Velocity (SV)* results.

Table 4.2: Descriptive statistics associated with *Sprint Velocity (SV)*.

	N	M	SD	Skew	Kurtosis
Scrum	10	26.8	2.3	-0.7	1.10
sMBSAP	9	31.8	2.2	-1.3	1.83

sMBSAP-driven sprint is presented in Figure 4.4. The lowest SV was 27 for Sprint 3, while the highest SV was 34 for Sprints 8 and 13. By comparison, the sMBSAP-driven sprints ($N = 9$) were associated with a numerically larger SV of $M = 31.8$ ($SD = 2.2$). The descriptive statistics are summarized in Table 4.2.

To test the hypothesis that the Scrum-driven and sMBSAP-driven sprints were associated with a statistically significantly different mean SV, an independent sample t-test was performed. A significance level of 0.05 was utilized. The Scrum-driven sprints and sMBSAP-driven sprints distributions were sufficiently normal for the purpose of conducting the t-test (i.e., $|\text{Skew}| < 2.0$ and $|\text{Kurtosis}| < 9.0$ [254]). Additionally, the required assumption on the homogeneity of variances was tested and satisfied via Levene's F test with $F(17) = 1.07$ and $p = 0.935$. Now, the independent sample t-test was found to be associated with a statistically significant effect: $t(17) = 4.78$ $\rightarrow p = 0.0002 \ll 0.05$. Thus, the sMBSAP-driven sprints were associated with a statistically significant larger mean SV than the Scrum-driven sprints. A graphical representation of the means and the 95% confidence intervals are displayed in Figure 4.4.

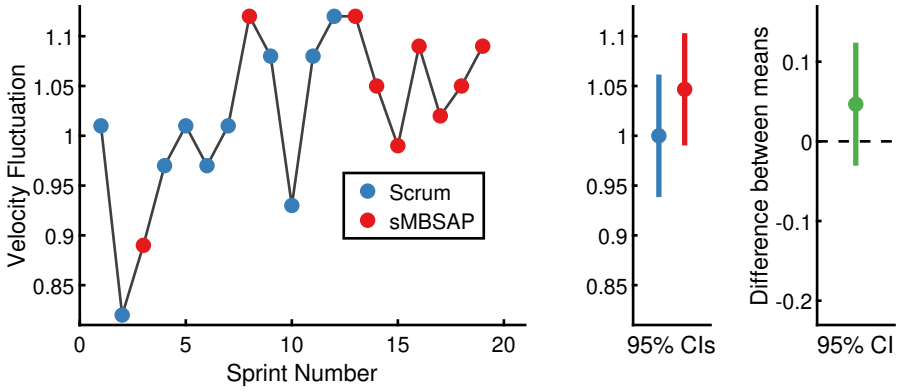


Figure 4.5: *Velocity Fluctuation* (VF) results.

Table 4.3: Descriptive statistics associated with *Velocity Fluctuation* (VF).

	N	M	SD	Skew	Kurtosis
Scrum	10	1.00	0.08	-0.75	1.16
sMBSAP	9	1.05	0.07	-1.26	1.71

4.2.2.2 Velocity Fluctuation (VF) Comparison

Based on the captured SV data, the VF of each sprint in chronological order is presented in Figure 4.5, calculated using Equation (3.5). For the Scrum-driven sprints, the lowest VF was 0.82 for Sprint 2, while the highest VF was 1.12 for Sprint 12. For the sMBSAP-driven sprints, the lowest VF was 0.89 for Sprint 3, while the highest VF was 1.12 for Sprints 8 and 13. The Scrum-driven sprints ($N = 10$) were associated with a VF standard deviation ($SD = 0.08$). By comparison, the sMBSAP-driven sprints ($N = 9$) were associated with a numerically very similar standard deviation of ($SD = 0.07$). The descriptive statistics are shown in Table 4.3.

The results from this study suggest no noticeable difference in the VF between the Scrum and sMBSAP-driven sprints.

4.2.2.3 Count of Lines of Code (CLOC) per Hour Comparison

The total CLOC written during the sprints for each approach was used as a secondary measure of *Productivity* and ended up being approximately 123,000 lines for the dietary recommendation system under consideration. The $CLOC$ written during the Scrum-driven sprints was 53,969, while the CLOC written during the sMBSAP-driven sprints was 69,679. The total de-

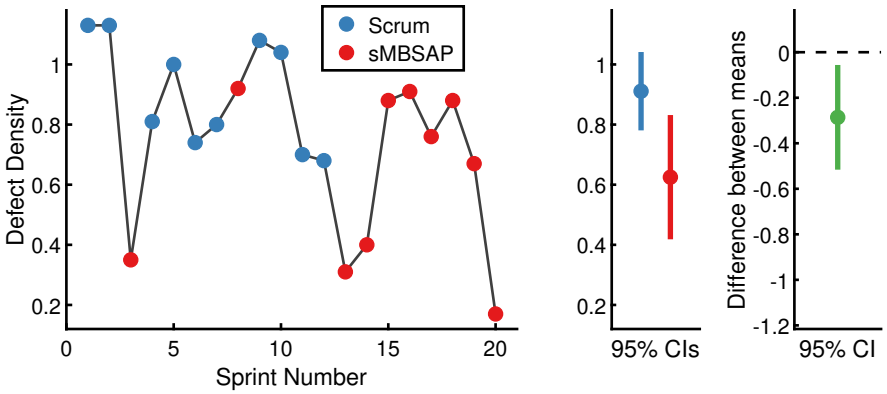


Figure 4.6: *Defect Density* (DD) results.

Table 4.4: Descriptive statistics associated with *Defect Density* (DD).

	N	M	SD	Skew	Kurtosis
Scrum	10	0.91	0.18	-0.009	-2.02
sMBSAP	10	0.63	0.29	-0.42	-1.7

velopment hours for the Scrum-driven sprints were 930 hours, while the total development hours for the sMBSAP-driven sprints were 980 hours. Accordingly, and using Equation (3.6), the $CLOC\ per\ hour$ for Scrum-driven sprints was 58.03. In the same manner, the $CLOC\ per\ hour$ for sMBSAP-driven sprints was 71.1. Thus, the sMBSAP-driven sprints were associated with a numerically larger $CLOC\ per\ hour$ than the Scrum-driven sprints.

4.2.3 Defect Rate Results

4.2.3.1 Defect Density (using PBIs) Comparison

Based on the captured defect data and Equation (3.7), the DD of each sprint is presented in Figure 4.6. For the Scrum-driven sprints, the lowest DD was 0.68 for Sprint 12, while the highest DD was 1.13 for Sprints 1 and 2. For the sMBSAP-driven sprints, the lowest DD was 0.17 for Sprint 20, while the highest DD was 0.92 for Sprint 8. The Scrum-driven sprints ($N = 10$) were associated with a DD of $M = 0.91$ ($SD = 0.18$). By comparison, the sMBSAP-driven sprints ($N = 10$) were associated with a numerically smaller DD of $M = 0.63$ ($SD = 0.29$). The descriptive statistics are shown in Table 4.4.

To test the hypothesis that the Scrum and sMBSAP-driven sprints were associated with a statistically significantly different mean DD, an independent sample t-test was performed. The unpaired t-test was used to compare the DD between Scrum-driven sprints and sMBSAP-driven sprints. An alpha level of 0.05 was utilized. The Scrum-driven sprints and sMBSAP-driven sprints distributions were sufficiently normal for the purpose of conducting the t-test (i.e., skew $< |2.0|$ and Kurtosis $< |9.0|$ [254]). Additionally, the assumption of homogeneity of variances was tested and satisfied via Levene's F test, $F(18) = 2.51, p = 0.1858$. The independent samples t-test was associated with a statistically significant effect, $t(18) = 2.64 \rightarrow p = 0.016 < 0.05$. Thus, the sMBSAP-driven sprints were associated with a statistically significant smaller mean DD than the Scrum-driven sprints. A graphical representation of the means and the 95% confidence intervals are displayed in Figure 4.6.

4.2.3.2 Defect Density (within Thousands (Kilo) of Lines of Code (KLOC)) Comparison

Based on the captured defect data and Equation (3.9), the DD of all Scrum-driven sprints (using KLOC) is 4.57. Similarly, the DD of all sMBSAP-driven sprints (using KLOC) is 2.24. The total CLOC written during the sprints for each approach was used as another measure of size [118]. The CLOC written for developing the dietary recommendation system is approximately 123,000. The CLOC written during the Scrum-driven sprints is 53,969, while the CLOC written during the sMBSAP-driven sprints is 69,679. The total number of defects detected during and after the Scrum-driven sprints is 247 defects, while the total defects detected during and after the sMBSAP-driven sprints are 156 defects. Accordingly, using Equation (3.9), the defects per KLOC for Scrum-driven sprints is 4.57. In the same manner, the defects per KLOC for sMBSAP-driven sprints is 2.24. Thus, the sMBSAP-driven sprints were associated with a numerically smaller DD using the KLOC as size than the Scrum-driven sprints.

4.2.3.3 Defect Leakage Comparisons

The DL for each sprint was calculated using Equation (3.10), and the results are shown in Figure 4.7. For the Scrum-driven sprints, the lowest DL was 16.7% for Sprint 1, while the highest

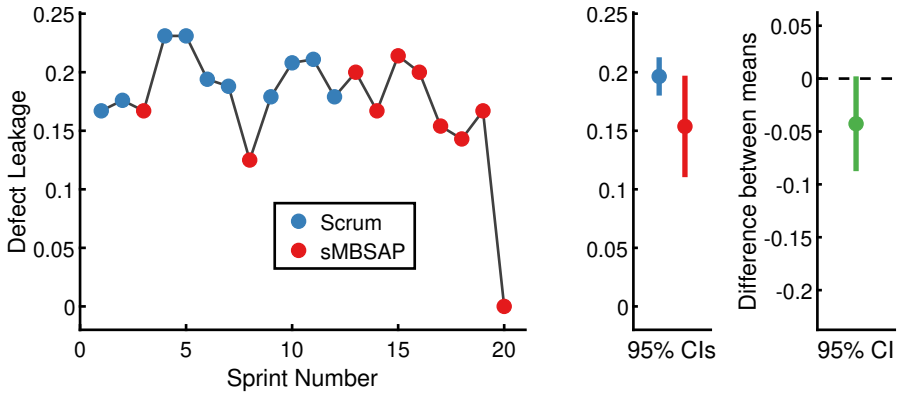


Figure 4.7: *Defect Leakage* (DL) results.

Table 4.5: Descriptive statistics associated with *Defect Leakage* (DL).

	N	M	SD	Skew	Kurtosis
Scrum	10	0.20	0.02	0.57	-1.13
sMBSAP	10	0.15	0.06	-2.09	5.31

DL was 23.1% for Sprints 4 and 5. Similarly, for sMBSAP-driven sprints, the lowest DL was 12.5% for Sprint 2, while the highest DL was 21.4% for Sprint 15. The Scrum-driven sprints ($N = 10$) were associated with a DL of $M = 0.2$ ($SD = 0.02$). By comparison, the sMBSAP-driven sprints ($N = 10$) were associated with a numerically smaller DL of $M = 0.15$ ($SD = 0.06$). The descriptive statistics are shown in Table 4.5.

To test the hypothesis that the Scrum and sMBSAP-driven sprints were associated with a statistically significantly different mean DL, an independent sample t-test was performed. The unpaired t-test was used to compare the DD between Scrum-driven sprints and sMBSAP-driven sprints. An alpha level of 0.05 was utilized. The Scrum-driven sprints and sMBSAP-driven sprints distributions were sufficiently normal for the purpose of conducting the t-test (i.e., skew $< |2.0|$ and Kurtosis $< |9.0|$ [254]). Additionally, the assumption of homogeneity of variances was tested and satisfied via Levene's F test, $F(18) = 2.51, p = 7.75$. The independent samples t-test was associated with a statistically significant effect, $t(18) = 2.13 \rightarrow p = 0.047 < 0.05$. Thus, the sMBSAP-driven sprints were associated with a statistically significant smaller mean DL than the Scrum-driven sprints. A graphical representation of the means and the 95% confidence intervals are displayed in Figure 4.7.

4.2.4 Other Observations

During *Sprint/Architecture Planning*, it was observed that the *Product Development Team* reached an agreement on the estimation of sMBSAP-driven sprints relatively faster than that of Scrum-driven sprints. The main difference in these meetings was the inputs presented to the team to discuss and conduct their estimation. sMBSAP used a system model as a single repository of the architecture and system *Requirements* [192]. Scrum used multiple tools to capture and present the architecture and system *Requirements*, such as ClickUp [163], Microsoft Office tools [255], and others [256]. The system model seems to have contributed to making the system architecture and *Requirements* more apparent, unambiguous, and easily communicated with the *Product Development Team* compared to those of the Scrum approach.

The MBSE tool [186] used during the experiment provided methods to create, retrieve, update, and delete the system *Requirements* and architecture content. The tool also aided in generating views of the architecture customized to the *Product Development Team* needs; the MBSE tool [186] also provided simulation capabilities of the architecture to support analysis and evaluation. In summary, the main benefit observed from using a model resided within its ability to visually represent the end-to-end software development flow, from user needs to delivered and supported software solutions.

It was also observed, during the *Daily Standups* and the *Sprint/Architecture Review*, that the *Product Development Team* demonstrated more grasp of the features and functions that need to be developed. Although both Scrum and sMBSAP aim at improving *Productivity*, the visual representation capabilities of sMBSAP gave it a comparative advantage.

The repeatable activities using architecture modeling and screen wireframing built-in within the model as the foundation contributed to unambiguous communication with the *Product Development Team*. The visually clear direction seems to have improved the capability of the *Product Development Team* to interpret the features and functions, accordingly, helping to complete more $StoryPoints(SP)$ within the same time frame and with fewer defects. Finally, using the Architecture model to highlight the completed parts of the system effectively communicated the progress

of the development effort. Similarly, using the Architecture model to highlight the parts of the system based on their *Defect Rate* effectively communicated the defect status.

4.3 Summary of the Results and Discussion

This chapter discusses the implementation of the sMBSAP approach and provides a side-by-side comparison of Scrum, MBSAP, and sMBSAP. Scrum and MBSAP share similarities, such as an emphasis on collaboration, iterative development, and continuous improvement. Both approaches value customer-centricity and prioritize delivering a working product. However, they have differences. Scrum prioritizes working software as progress, while MBSAP emphasizes using models to capture and communicate system information. sMBSAP is a middle point, using both formal and informal modeling languages to keep system information within a model. sMBSAP is primarily focused on software development, while MBSAP focuses on systems engineering and high-level system models. sMBSAP is application-agnostic and suitable for small to medium-sized projects. The chapter also discusses the key characteristics and benefits of implementing the sMBSAP approach.

This chapter also reports the results of the quasi-experiment. The average CR for Scrum-driven sprints was 0.81, while that of sMBSAP-driven sprints was 0.94. The average SV for Scrum-driven sprints was 26.8, while that of sMBSAP-driven sprints was 31.8. The CLOC $per\ hour$ for Scrum-driven sprints was 58.03, while that of sMBSAP-driven sprints was 71.1. The average DD for Scrum-driven sprints was 0.91, while that of sMBSAP-driven sprints was 0.63. The DD of all Scrum-driven sprints (using KLOC) was 4.57, while that of sMBSAP-driven sprints (using KLOC) was 2.24. The average DL for Scrum-driven sprints was 0.20, while that of sMBSAP-driven sprints was 0.15. The t-test results showed that the sMBSAP-driven sprints were associated with a statistically significant larger mean CR, SV, DD, and DL than that of the Scrum-driven sprints. No noticeable difference in the VF has been found between the Scrum and sMBSAP-driven sprints.

Chapter 5

Summary and Conclusions[5]

This chapter provides a summary of the Scrum Model Based System Architecture Process (sMBSAP) methodology (Section 5.1) and synthesis of the experiment results (Section 5.2). Then, provides conclusions of the research (Section 5.3). This chapter also sheds light on the research contribution (Section 5.4). Finally, it provides insight into the path forward in future work areas (Section 5.5) to better investigate the differences between Agile and Agile Model-Based Systems Engineering (MBSE) from the software development performance standpoint.

5.1 Summary of the Proposed sMBSAP Methodology

This research introduces a method to practically implement Agile MBSE called the Scrum Model Based System Architecture Process (sMBSAP) approach. The building blocks of sMBSAP includes processes, meetings, artifacts, and roles. The sMBSAP approach includes five main artifacts: *Product/Sprint Backlog*, *Operational Viewpoint (OV)*, *Logical/Functional Viewpoint (LV/FV)*, *Physical Viewpoint (PV)*, and *Product Increment*, as well as four roles: *Product Owner*, *Scrum Master*, *System Architect*, and *Product Development Team*.

sMBSAP relies on maintaining the system information and artifacts within the model to leverage the benefits of a centralized source of information. The five sMBSAP phases include Initiate, Plan and Architect, Implement, Review, and Retrospect, and Release. Although the experiment was conducted on a health technology software system, sMBSAP is application-agnostic and can be used with other software or engineered systems.

sMBSAP has some characteristics similar to those of Scrum and Model-Based System Architecture Process (MBSAP), including the focus on collaboration, iterative and incremental development, continuous improvement, customer-centricity, prioritizing delivering a working product to

[5]Parts of this chapter previously appeared in [2, 3].

Table 5.1: Summary of comparative analysis results.

Metric	Scrum	sMBSAP	Result
Commitment Reliability	0.81	0.94	sMBSAP has a statistically significant larger mean
Sprint Velocity	26.8	30.4	sMBSAP has a statistically significant larger mean
Velocity Fluctuation	0.08	0.07	No considerable difference
CLOC per Hour	58.03	71.1	sMBSAP has a numerically larger ratio
Defect Density (PBIs)	0.91	0.62	sMBSAP has a statistically significant smaller mean
Defect Density (KLOC)	4.57	2.24	sMBSAP has a numerically smaller ratio
Defect Leakage	19.6%	15.4%	sMBSAP has a statistically significant smaller mean

the customer, and responding to change. However, the sMBSAP, pragmatically, uses both formal and informal modeling languages to keep system information within the model. The level of details of the system architecture can be decided by the *Product Development Team* to fit the needs and the size of the project, which may range from small to large.

5.2 Synthesis of Results

In this experiment, the *Reliability of Estimation*, *Productivity*, and *Defect Rate* metrics we compared for both the Scrum-driven and sMBSAP-driven sprints. The comparison of *Reliability of Estimation* is conducted based on the results of the CR. The comparisons of *Productivity* are conducted based on the results of the SV, VF, and $CLOC\ per\ hour$. Finally, the comparisons of *Defect Rate* are conducted based on the results of the DD and DL. Table 5.1 summarizes the statistical analysis results.

The purpose of this research was to investigate whether there are differences in the system development performance between sMBSAP and Scrum.

Comparing sMBSAP and Scrum, are there measurable benefits to system development performance when using one approach over the other while developing a health technology system?

To address this question, three software development performance objectives were investigated using the development of a health technology system as an experiment. Three hypotheses were developed, one for each software development performance objective. The results of each of the three hypotheses are below.

Results of First Hypothesis. To test Hypothesis 1, a t-test was conducted to compare the *Reliability of Estimation* between sMBSAP and Scrum. The analysis revealed a statistically significant difference in CR between the two groups. Therefore, H_1 for the first Hypothesis is rejected, indicating that there is indeed a significant difference in CR between the treatment group (sMBSAP) and the control group (Scrum).

Results of Second Hypothesis. To test Hypothesis 2, (a) a t-test was conducted to compare the *Productivity* between sMBSAP and Scrum. The analysis revealed a statistically significant difference in SV between the two groups. Therefore, H_1 for the second Hypothesis is rejected, indicating that there is indeed a significant difference in SV between the treatment group (sMBSAP) and the control group (Scrum). (b) Simiralirly, a t-test was conducted to compare VF. The analysis suggests no noticeable difference in the VF between the Scrum and sMBSAP-driven sprints. Therefore, H_1 for the second Hypothesis is rejected, indicating the VF of the treatment group (sMBSAP) is not significantly different than that of the control group (Scrum). (c) By calculating the $CLOC\ per\ hour$. It was found that the sMBSAP-driven sprints were associated with a numerically larger $CLOC\ per\ hour$ than the Scrum-driven sprints. Therefore, H_1 for the second Hypothesis is rejected, indicating the $CLOC\ per\ hour$ of the treatment group (sMBSAP) is numerically larger than that of the control group (Scrum).

Results of Third Hypothesis. To test Hypothesis 3, (a) a t-test was conducted to compare the DD between sMBSAP and Scrum. The analysis revealed a statistically significant difference in DD (using Product Backlog Items (PBIs)) between the two groups. Therefore, H_1 for the third Hypothesis is rejected, indicating that there is indeed a significant difference in DD (using PBIs)

between the treatment group (sMBSAP) and the control group (Scrum). By calculating the DD (using KLOC). It was found that the sMBSAP-driven sprints were associated with a numerically smaller DD (using KLOC) than the Scrum-driven sprints. Therefore, H_1 for the third Hypothesis is rejected, indicating the DD (using KLOC) of the treatment group (sMBSAP) is numerically smaller than that of the control group (Scrum). (b) Similarly, a t-test was conducted to compare VF. The analysis suggests no noticeable difference in the VF between the Scrum and sMBSAP-driven sprints. Therefore, H_1 for the third Hypothesis is rejected, indicating the VF of the treatment group (sMBSAP) is not significantly different than that of the control group (Scrum). (b) a t-test was conducted to compare the DL between sMBSAP and Scrum. The analysis revealed a statistically significant difference in DL between the two groups. Therefore, H_1 for the third Hypothesis is rejected, indicating that there is indeed a significant difference in DL between the treatment group (sMBSAP) and the control group (Scrum).

5.3 Conclusions

In the first part of this research, an integration of the Agile and MBSE approaches has been proposed. The new approach, termed the Scrum Model Based System Architecture Process (sMBSAP), uses the same cyclic approach of Scrum and the MBSAP. The sMBSAP approach includes five main artifacts: *Product/Sprint Backlog*, *OV*, *LV/FV*, *PV*, and *Product Increment*, as well as four roles: *Product Owner*, *Scrum Master*, *System Architect*, and *Product Development Team*. The five sMBSAP phases include Initiate, Plan and Architect, Implement, Review and Retrospect, and Release.

Both Scrum and the MBSAP focus on collaboration and continuous improvement. Both approaches also value customer-centricity and prioritize delivering a working product to the customer. These similarities have been passed down to the sMBSAP. The sMBSAP customizes the artifacts to keep system information within the model. The sMBSAP is application agnostic and can be applied to software or other industries. As for project size, the sMBSAP can be customized

to fit small to large projects. The sMBSAP approach was validated through a pilot study to develop a health technology system over one year.

The preliminary results have shown that the proposed approach contributed to achieving the desired system development outcomes and, at the same time, generated complete system architecture artifacts that would not have been developed if Agile alone had been used. The highlights of the sMBSAP approach and benefits observed during the implementation can be summarized as follows: (1) The *System Architect* works closely, not in a silo, with the *Product Development Team* to customize, empower, and educate the team to get the best out of the architecture model; (2) selecting an MBSE tool with built-in methodologies and models helps create a faster first iteration of the model; (3) the sMBSAP enables the *System Architect* to use Agile terminologies that the *Product Development Team* understands; (4) the MBSE tool enables the *System Architect* to combine formal and informal modeling to gradually shift the mindset of the Agile team towards MBSE; (5) the architecture effort progresses one sprint at a time; (6) the *Requirements* and *User Stories* will be traced to the various perspectives of the system model throughout the following iterations or viewpoints; (7) the sMBSAP is a practical middle ground between light documentation enthusiasts and those who value heavy documentation.

The promising results observed while using the model are a step towards closing the gap between Agile and MBSE. The sMBSAP offered a practical and operational method for achieving the desired and potentially better outcomes compared to either approach alone. In parallel, this research shows that the sMBSAP is more aligned with federal and state regulations, which promote Agile in its systems engineering guidelines while requiring a proper set of system documentation.

The purpose of this study was to compare the software development performance of sMBSAP vs. that of Scrum. Investigating the relative value of an Agile MBSE approach over Agile alone might solve one of the obstacles to the adoption of Agile MBSE. Towards this goal, results were collected from a quasi-experimental posttest-only with nonequivalent groups research design for the software development of a health technology system. The independent variables were the two software development methods, sMBSAP and Scrum, actuated during different development

sprints. The dependent variables are the software system development performance objectives of (1) *Reliability of Estimation* as measured by CR; (2) *Defect Rate* as measured by DD using PBIs, DD using KLOC and DL; (3) *Productivity* as measured by SV, VF, and $CLOC\ per\ hour$. A total of twenty sprints were executed, with ten sprints executed for each approach, respectively.

From the results, the observed *Commitment Reliability* CR and *Productivity* for the sMBSAP-driven sprints was larger than that of Scrum-driven sprints; and the observed *Defect Rate* for the sMBSAP-driven sprints was smaller than that of Scrum-driven sprints. The improved *Reliability of Estimation*, *Productivity*, and *Defect Rate* could potentially help reduce the risk of running behind schedule and over budget that can occur with Agile-driven projects. Overall, these results provide some evidence of the efficacy of a combined Agile MBSE approach in managing software-based systems and strengthen the case for its adoption within the software development community and the broader systems engineering community.

The factors that pose threats to reliability, validity, and statistical conclusion validity have been identified, monitored, and mitigated to minimize the impact of these factors on the quality of the research design. However, there are still aspects of the research design that could be improved to provide additional clear and convincing evidence on the relative value of Agile MBSE approaches.

Finally, it is important to note that the using MBSE approaches, in general, is associated with more time and expertise required to architect the system and leverage the full potential of the model. Taking advantage of advanced capabilities, available in MBSE tools, such as simulation, requires even more time and training.

5.4 Research Contributions

The main contribution of this research lies in introducing a practical and operational method for merging Agile and MBSE. In parallel, the results suggest that sMBSAP is a middle ground that is more aligned with federal and state regulations, as it addresses the technical debt concerns.

This research also fills a gap in the literature through an experimental study providing side-by-side comparisons of sMBSAP and Scrum to describe and develop a health technology system with

promising quantitative results of the relative advantages of an Agile MBSE approach over Agile. The comparison is conducted for three widely used software development metrics: *Reliability of Estimation*. *Productivity* and *Defect Rate*. Finally, the research produced an architecture framework for describing a health technology system that defines the *Structure*, *Data*, *Behavior*, and *Capabilities* at the system and subsystem levels.

5.5 Future Work

Due to several practical limitations, this study did not pursue a true experimental design (and this is likely the case for many software development research activities). However, a potential direction may include developing the same product in two parallel tracks (i.e., developing the same product twice); one track using an Agile MBSE and the other using an Agile approach. The data collected from such an experiment would be statistically paired rather than unpaired. Moreover, the software development objectives will be measured for the same set of functions or *Product Backlog Items*, potentially eliminating the confounding variable of experimenting on 2 different sets of functions. Even if a single track might be used (i.e., developing the product once, as was done in this study), the researcher might consider a true experimental design with random sampling from a given population when assigning an approach to a given sprint. However, care must be taken to ensure that such a nontraditional development plan does not adversely impact the momentum and other factors.

Based on the results of this research, there is a need to conduct more comparative analyses between Agile and Agile MBSE methods using other software development objectives, techniques, and metrics (e.g., using a technique other than Planning Poker in estimation or size metrics other than Story Points). Conducting a comparative analysis for software products in industries other than health technology, larger settings, and against Agile methodologies other than Scrum would also provide additional insights for the software development community. Future investigations may also test sMBSAP with non-software systems to validate the methodology across other dis-

ciplines. One of the opportunities worth investigating is the generation of executable code from implementation-level models and its effect on productivity and reusability.

Another important consideration that requires further investigation is examining the influence of *Requirements* volatility on the *Defect Rate*. Future lines of questioning may also consider testing the sMBSAP methodology while considering the impact of soft or human-related factors on software development *Productivity*. Finally, investigating the implementation of the sMBSAP methodology while considering the non-technical tasks consumed during the sprints would be an interesting extension.

Bibliography

[1] J. M. Borky and T. H. Bradley, *Effective Model-Based Systems Engineering*. Springer, 2019, doi: 10.1007/978-3-319-95669-5.

[2] M. Huss, D. R. Herber, and J. M. Borky, "An Agile model-based software engineering approach illustrated through the development of a health technology system," *Software*, vol. 2, no. 2, pp. 234–257, 2023, doi: 10.3390/software2020011.

[3] ——, "Comparing measured agile software development metrics using an agile model-based software engineering approach versus scrum only," *Software*, vol. 2, no. 3, pp. 310–331, 2023, doi: 10.3390/software2030015.

[4] H. Tohidi, "The role of risk management in IT systems of organizations," *Procedia Computer Science*, vol. 3, pp. 881–887, 2011, doi: 10.1016/j.procs.2010.12.144.

[5] Standish Group, "CHAOS Report 2020," Standish Group, Tech. Rep., 2020.

[6] Y. Hooshmand, D. Adamenko, S. Kunnen, and P. Köhler, "An approach for holistic model-based engineering of industrial plants," in *International Conference on Engineering Design*, vol. 3, Vancouver, Canada, Aug. 2017, pp. 101–110.

[7] S. Kleiner, S. Husung, S. Schulze, C. Tschirner, and R. Kaffenberger, "Model based systems engineering: Prinzipen, anwendung, beispiele, erfahrung und nutzen aus praxissicht," *Tag des Systems Engineering*, pp. 13–22, 2016.

[8] E. Kindler, "Model-based software engineering and process-aware information systems," in *Transactions on Petri Nets and Other Models of Concurrency II*. Springer, 2009, pp. 27–45.

[9] S. Friedenthal, A. Moore, and R. Steiner, *A Practical Guide to SysML: The Systems Modeling Language*, 3rd ed. Morgan Kaufmann, 2015, doi: 10.1016/C2013-0-14457-1.

[10] N. M. J. Basha, S. A. Moiz, and M. Rizwanullah, "Model based software development: issues & challenges," *International Journal of Computer Science and Informatics*, vol. 3, no. 2, pp. 84–88, 2013, doi: 10.47893/IJCSI.2013.1123.

[11] D. R. Call and D. R. Herber, "Applicability of the diffusion of innovation theory to accelerate model-based systems engineering adoption," *Systems Engineering*, vol. 25, no. 6, pp. 574–583, Nov. 2022, doi: 10.1002/sys.21638.

[12] S. Y. Kim, D. Wagner, and A. Jimenez, "Challenges in applying model-based systems engineering: human-centered design perspective," in *INCOSE Human-Systems Integration Conference*, 2019.

[13] M. Chami, A. Aleksandraviciene, A. Morkevicius, and J.-M. Bruel, "Towards solving mbse adoption challenges: the d3 mbse adoption toolbox," in *INCOSE International Symposium*, vol. 28, no. 1. Wiley Online Library, 2018, pp. 1463–1477.

[14] M. Chami and J.-M. Bruel, "A survey on mbse adoption challenges," 2018.

[15] V. Salehi, G. Florian, J. Taha *et al.*, "Implementation of systems modeling language (sysml) in consideration of the consens approach," in *DS 92: Proceedings of the DESIGN 2018 15th International Design Conference*, 2018, pp. 2987–2998.

[16] "ISO/IEC/IEEE 15288:2023 systems and software engineering – system life cycle processes," ISO, IEC, and IEEE, Standard, May 2023, doi: 10.1109/IEEESTD.2023.10123367.

[17] R. Dove and W. Schindel, "Agility in systems engineering – findings from recent studies," Paradigm Shift International and ICTT System Sciences, Working Paper, Nov. 2017.

[18] R. Turner, "Toward agile systems engineering processes," *Crosstalk. The Journal of Defense Software Engineering*, pp. 11–15, 2007.

[19] U. A. Altahtooh and M. W. Emsley, "Is a challenged project one of the final outcomes for an it project?" in *2014 47th Hawaii International Conference on System Sciences*, 2014, pp. 4296–4304, doi: 10.1109/HICSS.2014.531.

[20] N. Muganda Ochara, J. Kandiri, and R. Johnson, "Influence processes of implementation effectiveness in challenged information technology projects in Africa," *Information Technology & People*, vol. 27, no. 3, pp. 318–340, Jan. 2014, doi: 10.1108/ITP-09-2013-0167.

[21] K. T. Yeo, "Critical failure factors in information system projects," *International Journal of Project Management*, vol. 20, no. 3, pp. 241–246, Apr. 2002, doi: 10.1016/S0263-7863(01)00075-8.

[22] V. Salehi and S. Wang, "Munich Agile MBSE concept (MAGIC)," *Proceedings of the Design Society: International Conference on Engineering Design*, vol. 1, no. 1, pp. 3701–3710, Jul. 2019, doi: 10.1017/dsi.2019.377.

[23] M. Riesener, C. Doelle, S. Perau, P. Lossie, and G. Schuh, "Methodology for iterative system modeling in Agile product development," *Procedia CIRP*, vol. 100, pp. 439–444, Jan. 2021, doi: 10.1016/j.procir.2021.05.101.

[24] M. Bott and B. Mesmer, "An analysis of theories supporting Agile scrum and the use of scrum in systems engineering," *Engineering Management Journal*, vol. 32, no. 2, pp. 76–85, 2020, doi: 10.1080/10429247.2019.1659701.

[25] B. P. Douglass, *Agile Model-Based Systems Engineering Cookbook*, 1st ed. Packt Publishing, Mar. 2021.

[26] P. D. Ciampa and B. Nagel, "Accelerating the Development of Complex Systems in Aeronautics via MBSE and MDAO: A Roadmap to Agility," in *AIAA Aviation Forum*, no. AIAA 2021-3056, Aug. 2021, doi: 10.2514/6.2021-3056.

[27] E. Bouillon, B. Güldali, A. Herrmann, T. Keuler, D. Moldt, and M. Riebisch, "Leichtgewichtige traceability im agilen entwicklungsprozess am beispiel von scrum," *Softwaretechnik-Trends*, vol. 33, no. 1, pp. 29–30, 2013.

[28] T. Lethbridge, J. Singer, and A. Forward, "How software engineers use documentation: the state of the practice," *IEEE Software*, vol. 20, no. 6, pp. 35–39, 2003, doi: 10.1109/MS.2003.1241364.

[29] S. Voigt, D. Hüttemann, A. Gohr, and M. Große, "Agile documentation tool concept," in *Developments and Advances in Intelligent Systems and Applications*. Springer, 2018, pp. 67–79.

[30] K. Schwaber, *Agile Project Management with Scrum*. Microsoft Press, 2004.

[31] W. J. Orlikowski, "Using Technology and Constituting Structures: A Practice Lens for Studying Technology in Organizations," *Organization Science*, vol. 11, no. 4, pp. 404–428, Aug. 2000, doi: 10.1287/orsc.11.4.404.14600.

[32] M. Bloch, S. Blumberg, and J. Laartz, "Delivering large-scale it projects on time, on budget, and on value," *Harvard Business Review*, vol. 5, no. 1, pp. 2–7, 2012.

[33] B. Flyvbjerg and A. Budzier, "Why Your IT Project May Be Riskier Than You Think," *Harvard Business Review*, Sep. 2011.

[34] "The 20 Most Successful Tech Failures of All Time," http://time.com, 2017.

[35] R. Lehavy and S. Udpa, "Kmart: Predicting Bankruptcy, Fresh Start Reporting, and Valuation of Distressed Securities," *Issues in Accounting Education*, vol. 26, no. 2, pp. 391–419, May 2011, doi: 10.2308/iace-10017.

[36] N. Heath, "Ten budget-busting IT disasters you should learn from," Oct. 2015.

[37] "What Can the State Do to Prevent Future IT Project Delays and Cost Overruns," Room 444, p. 9, 2015.

[38] N. B. Ruparelia, "Software development lifecycle models," *ACM SIGSOFT Software Engineering Notes*, vol. 35, no. 3, pp. 8–13, May 2010, doi: 10.1145/1764810.1764814.

[39] M. A. Rather and M. V. Bhatnagar, "A comparative study of software development life cycle models," *International Journal of Application or Innovation in Engineering & Management*, vol. 4, no. 10, pp. 23–29, Oct. 2015, doi: 10.2648/IJAIEM.691.1423.

[40] B.-Y. Tsai, S. Stobart, N. Parrington, and B. Thompson, "Iterative design and testing within the software development life cycle," *Software Quality Journal*, vol. 6, no. 4, pp. 295–310, 1997, doi: 10.1023/A:1018528506161.

[41] A. Kossiakoff, S. M. Biemer, S. J. Seymour, and D. A. Flanigan, *Systems engineering principles and practice*. John Wiley & Sons, 2020.

[42] S. K. Dora and P. Dubey, "Software development life cycle (sdlc) analytical comparison and survey on traditional and Agile methodology," *National Monthly Refereed Journal of Research in Science & Technology*, vol. 2, no. 8, pp. 22–30, 2013.

[43] L. Khong, L. Yu Beng, T. Yip, and T. Soofun, "Software development life cycle AGILE vs traditional approaches," in *International Conference on Information and Network Technology*, vol. 37, Feb. 2012, pp. 162–167.

[44] B. Boehm, "Get ready for agile methods, with care," *Computer*, vol. 35, no. 1, pp. 64–69, 2002, doi: 10.1109/2.976920.

[45] M. Javanmard and M. Alian, "Comparison between Agile and Traditional software development methodologies," *Cumhuriyet Üniversitesi Fen Edebiyat Fakültesi Fen Bilimleri Dergisi*, vol. 36, no. 3, pp. 1386–1394, May 2015.

[46] C. Quist, "Benefits of Blending Agile and Waterfall Project Planning Methodologies," University of Oregon, Tech. Rep., Dec. 2015.

[47] L. Williams and A. Cockburn, "Agile software development: It's about feedback and change," *Computer*, vol. 36, no. 6, pp. 39–43, Jun. 2003, doi: 10.1109/MC.2003.1204373.

[48] A. Dennis, B. H. Wixom, and D. Tegarden, *Systems Analysis and Design with UML Version 2.0: An Object-Oriented Approach*, 4th ed. John Wiley & Sons, 2005.

[49] S. R. Schach, "Introduction to Object-Oriented Systems Analysis and Design with Uml and the Unified Process," *undefined*, Jul. 2003.

[50] L. Constantine and L. Lockwood, "Usage-centered software engineering: An agile approach to integrating users, user interfaces, and usability into software engineering practice," in *25th International Conference on Software Engineering, 2003. Proceedings.*, May 2003, pp. 746–747, doi: 10.1109/ICSE.2003.1201267.

[51] R. T. Watson, G. G. Kelly, R. D. Galliers, and J. C. Brancheau, "Key Issues in Information Systems Management: An International Perspective," *Journal of Management Information Systems*, vol. 13, no. 4, pp. 91–115, Mar. 1997, doi: 10.1080/07421222.1997.11518144.

[52] A. Mishra and Y. I. Alzoubi, "Structured software development versus agile software development: a comparative analysis," *International Journal of System Assurance Engineering and Management*, pp. 1–19, 2023, doi: 10.1007/s13198-023-01958-5.

[53] C. Schmidt, *Agile Software Development Teams*, ser. Progress in IS. Springer, 2015, doi: 10.1007/978-3-319-26057-0.

[54] Manifesto for agile software development. url: http://agilemanifesto.org.

[55] S. Sharma, D. Sarkar, and D. Gupta, "Agile processes and methodologies: A conceptual study," *International journal on computer science and Engineering*, vol. 4, no. 5, p. 892, 2012.

[56] K. Schwaber, "Scrum development process," in *Business object design and implementation*. Springer, 1997, pp. 117–134.

[57] L. R. Vijayasarathy and D. Turk, "Agile software development: A survey of early adopters," *Journal of Information Technology Management*, vol. XIX, no. 2, pp. 1–8, 2008.

[58] U.S. Government Accountability Office, "Agile assessment guide: Best practices for agile adoption and implementation," Washington, DC, Tech. Rep. GAO-20-590G, Sep.

[59] E. R. Carroll and R. J. Malins, "Systematic literature review: how is model-based systems engineering justified?" Sandia National Laboratories, Tech. Rep. SAND2016-2607, Mar. 2016, doi: 10.2172/1561164.

[60] D. Azhar, E. Mendes, and P. Riddle, "A systematic review of web resource estimation," in *International Conference on Predictive Models in Software Engineering*, 2012, pp. 49–58, doi: 10.1145/2365324.2365332.

[61] M. Jorgensen and M. Shepperd, "A systematic review of software development cost estimation studies," *IEEE Transactions on Software Engineering*, vol. 33, no. 1, pp. 33–53, 2006, doi: 10.1109/TSE.2007.256943.

[62] K. Molokken and M. Jorgensen, "A review of software surveys on software effort estimation," in *International Symposium on Empirical Software Engineering*, 2003, pp. 223–230, doi: 10.1109/ISESE.2003.1237981.

[63] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," *Information and Software Technology*, vol. 54, no. 1, pp. 41–59, 2012, doi: 10.1016/j.infsof.2011.09.002.

[64] M. Cohn, *Agile Estimating and Planning*. Pearson Education, 2005.

[65] J. Grenning, "Planning poker or how to avoid analysis paralysis while release planning," Hawthorn Woods: Renaissance Software Consulting, White Paper, 2002.

[66] V. Mahnič and T. Hovelja, "On using planning poker for estimating user stories," *Journal of Systems and Software*, vol. 85, no. 9, pp. 2086–2095, 2012, doi: 10.1016/j.jss.2012.04.005.

[67] J. Dalton, "Planning poker," in *Great Big Agile*. Springer, 2019, pp. 203–204, doi: 10.1007/978-1-4842-4206-3_44.

[68] N. C. Haugen, "An empirical study of using planning poker for user story estimation," in *AGILE*, 2006, doi: 10.1109/AGILE.2006.16.

[69] K. Moløkken-Østvold, N. C. Haugen, and H. C. Benestad, "Using planning poker for combining expert estimates in software projects," *Journal of Systems and Software*, vol. 81, no. 12, pp. 2106–2117, 2008, doi: 10.1016/j.jss.2008.03.058.

[70] K. Molokken-Ostvold and N. C. Haugen, "Combining estimates with planning poker–an empirical study," in *Australian Software Engineering Conference*, 2007, pp. 349–358, doi: 10.1109/ASWEC.2007.15.

[71] K. Moløkken-Østvold, M. Jørgensen, S. S. Tanilkan, H. Gallis, A. C. Lien, and S. W. Hove, "A survey on software estimation in the Norwegian industry," in *International Symposium on Software Metrics, 2004. Proceedings.*, 2004, pp. 208–219, doi: 10.1109/METRIC.2004.1357904.

[72] S. Grimstad and M. Jørgensen, "A framework for the analysis of software cost estimation accuracy," in *ACM/IEEE International Symposium on Empirical Software Engineering*, Sep. 2006, pp. 58–65, doi: 10.1145/1159733.1159745.

[73] K. M. Furulund and K. Molkken-stvold, "Increasing Software Effort Estimation Accuracy Using Experience Data, Estimation Models and Checklists," in *Seventh International Conference on Quality Software (QSIC 2007)*, Oct. 2007, pp. 342–347, doi: 10.1109/QSIC.2007.4385518.

[74] M. Tiwari, "Understanding Scrum metrics and KPIs," Online, Nov. 2019, medium. url: https://medium.com/@techiemanoj4/understanding-scrum-metrics-and-kpis-fcf56833d488.

[75] J. Smart, "To transform to have agility, dont do a capital A, capital T Agile transformation," *IEEE Software*, vol. 35, no. 6, pp. 56–60, 2018, doi: 10.1109/MS.2018.4321245.

[76] M. Khanzadi, M. M. Shahbazi, and H. Taghaddos, "Forecasting schedule reliability using the reliability of agents' promises," *Asian Journal of Civil Engineering*, vol. 19, pp. 949–962, 2018, doi: 10.1007/s42107-018-0075-7.

[77] C. Chen, S. Housley, P. Sprague, and P. Goodlad, "Introducing lean into the UK Highways Agency's supply chain," *Proceedings of the Institution of Civil Engineers - Civil Engineering*, vol. 165, no. 5, pp. 34–39, 2012, doi: 10.1680/cien.11.00013.

[78] Project Management Institute, *A Guide to the Project Management Body of Knowledge*, 4th ed. Newtown Square, Pa: Project Management Inst, Jan. 2009.

[79] F. Anbari, "Earned Value Project Management Method and Extensions," *Project Management Journal*, vol. 34, pp. 12–23, Dec. 2003, doi: 10.1177/875697280303400403.

[80] W. F. Abba, "Earned Value Management — Reconciling Government and Commercial Practices For People Involved in Earned Value — Government, Industry, Academia, Or Consulting — These Are Exciting Times!" *undefined*, 1997.

[81] D. S. Christensen, "A review of cost/schedule control systems criteria literature," *Project Management Journal*, vol. 25(3), 32–39, 54, 1994.

[82] E. Kim, "A study on the effective implementation of earned value management methodology /," Jan. 2000.

[83] E. Kim, W. G. Wells, and M. R. Duffey, "A model for effective implementation of Earned Value Management methodology," *International Journal of Project Management*, vol. 21, no. 5, pp. 375–382, Jul. 2003, doi: 10.1016/S0263-7863(02)00049-2.

[84] Q. W. Fleming and J. M. Koppelman, "Earned value project management." Project Management Institute, 2016.

[85] S. Sackey, D.-E. Lee, and B.-S. Kim, "Duration estimate at completion: Improving earned value management forecasting accuracy," *KSCE Journal of Civil Engineering*, vol. 24, pp. 693–702, 2020.

[86] A. Balali, A. Valipour, J. Antuchevičienė, and J. Šaparauskas, "Improving the Results of the Earned Value Management Technique Using Artificial Neural Networks in Construction Projects," 2020.

[87] S. K. Bhosekar and G. Vyas, "Cost Controlling Using Earned Value Analysis in Construction Industries," *International Journal of Engineering and Innovative Technology*, vol. 1, no. 4, pp. 324–332, Apr. 2012.

[88] T. C. Keng and N. Shahdan, "THE APPLICATION OF EARNED VALUE MANAGEMENT (EVM) IN CONSTRUCTION PROJECT MANAGEMENT," *Journal of Technology Management and Business*, vol. 2, no. 2, Dec. 2015.

[89] V. Aramali, H. Sanboskani, G. E. Gibson, M. El Asmar, and N. Cho, "Forward-Looking State-of-the-Art Review on Earned Value Management Systems: The Disconnect between Academia and Industry," *Journal of Management in Engineering*, vol. 38, no. 3, p. 03122001, May 2022, doi: 10.1061/(ASCE)ME.1943-5479.0001019.

[90] A. Nizam, H. M. Aburas, and A. K. Elshennawy, "Earned Value in a Project Manufacturing Environment - A Case Study Assessing the Effectiveness of EVM," *The Journal of Modern Project Management*, vol. 8, no. 2, 2020, doi: 10.19255/JMPM02404.

[91] R. S. Carson and B. Zlicaric, "Using Performance-Based Earned Value® for Measuring Systems Engineering Effectiveness," *INCOSE International Symposium*, vol. 18, no. 1, pp. 111–122, Jun. 2008, doi: 10.1002/j.2334-5837.2008.tb00794.x.

[92] V. Kadary, "On application of earned value index to software productivity metrics in embedded computer systems," in *1992 Proceedings Computer Systems and Software Engineering*.

IEEE Computer Society, Jan. 1992, pp. 666,667,668,669,670–666,667,668,669,670, doi: 10.1109/CMPEUR.1992.218455.

[93] A. Cabri and M. Griffiths, "Earned value and agile reporting," in *AGILE 2006 (AGILE'06)*, Jul. 2006, pp. 6 pp.–22, doi: 10.1109/AGILE.2006.21.

[94] J. Kantor, K. Long, J. Becla, F. Economou, M. Gelman, M. Juric, R. Lambert, S. Krughoff, J. D. Swinbank, and X. Wu, "Agile software development in an earned value world: A survival guide," in *Modeling, Systems Engineering, and Project Management for Astronomy VII*, vol. 9911. SPIE, Aug. 2016, pp. 261–278, doi: 10.1117/12.2233380.

[95] D. Kuchta, "Combination of the earned value method and the agile approach–a case study of a production system implementation," in *Intelligent Systems in Production Engineering and Maintenance*, 2019, pp. 87–96, doi: 10.1007/978-3-319-97490-3_9.

[96] K. Manship, "Implementing Earned Value Management on Agile Projects," University of Oregon, Tech. Rep., 2018.

[97] T. Sulaiman, B. Barton, and T. Blackburn, "AgileEVM - earned value management in Scrum Projects," in *AGILE 2006 (AGILE'06)*, Jul. 2006, pp. 10 pp.–16, doi: 10.1109/AGILE.2006.15.

[98] E. S. F. Cardozo, J. B. F. Araújo Neto, A. Barza, A. C. C. França, and F. Q. B. da Silva, "Scrum and productivity in software projects: a systematic literature review," in *International Conference on Evaluation and Assessment in Software Engineering*, Apr. 2010, doi: 10.14236/ewic/EASE2010.16.

[99] S. Wagner and M. Ruhe, "A systematic review of productivity factors in software development," arXiv, 2018, doi: 10.48550/arXiv.1801.06475.

[100] K. Maxwell and P. Forselius, "Benchmarking software development productivity," *IEEE Software*, vol. 17, no. 1, pp. 80–88, 2000, doi: 10.1109/52.820015.

[101] F. C. Brodbeck, *Produktivität und Qualität in Software-Projekten: psychologische Analyse und Optimierung von Arbeitsprozessen in der Software-Entwicklung*. Oldenbourg, 1994.

[102] B. Kitchenham and E. Mendes, "Software productivity measurement using multiple size measures," *IEEE Transactions on Software Engineering*, vol. 30, no. 12, pp. 1023–1035, Dec. 2004, doi: 10.1109/TSE.2004.104.

[103] H. Huijgens and R. v. Solingen, "A replicated study on correlating agile team velocity measured in function and story points," in *International Workshop on Emerging Trends in Software Metrics*, 2014, pp. 30–36, doi: 10.1145/2593868.2593874.

[104] G. B. Alleman, M. Henderson, and R. Seggelke, "Making Agile development work in a government contracting environment-measuring velocity with earned value," in *Agile Development Conference*, Jun. 2003, pp. 114–119, doi: 10.1109/ADC.2003.1231460.

[105] T. Javdani, H. Zulzalil, A. A. A. Ghani, A. B. M. Sultan, and R. M. Parizi, "On the current measurement practices in agile software development," arXiv, 2013, doi: 10.48550/arXiv.1301.5964.

[106] E. Kupiainen, M. V. Mäntylä, and J. Itkonen, "Using metrics in Agile and lean software development–a systematic literature review of industrial studies," *Information and Software Technology*, vol. 62, pp. 143–163, 2015, doi: 10.1016/j.infsof.2015.02.005.

[107] T. Killalea, "Velocity in software engineering," *Communications of the ACM*, vol. 62, no. 9, pp. 44–47, 2019, doi: 10.1145/3345626.

[108] S. Sharma, D. Kumar, and M. E. Fayad, "An impact assessment of agile ceremonies on sprint velocity under agile software development," in *International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)*, 2021, doi: 10.1109/ICRITO51393.2021.9596508.

[109] F. Albero Pomar, J. A. Calvo-Manzano, E. Caballero, and M. Arcilla-Cobián, "Understanding sprint velocity fluctuations for improved project plans with scrum: a case study,"

Journal of Software: Evolution and Process, vol. 26, no. 9, pp. 776–783, 2014, doi: 10.1002/smr.1661.

[110] "IEEE standard glossary of software engineering terminology," IEEE, Standard, 1990, doi: 10.1109/IEEESTD.1990.101064.

[111] P. Tripathy and K. Naik, *Software Testing and Quality Assurance: Theory and Practice*. John Wiley & Sons, 2011, doi: 10.1002/9780470382844.

[112] M. McDonald, R. Musson, and R. Smith, *The Practical Guide to Defect Prevention*. Microsoft Press, 2007.

[113] C. Jones and O. Bonsignour, *The Economics of Software Quality*. Addison-Wesley Professional, 2011.

[114] B. Boehm and V. R. Basili, "Top 10 list [software development]," *Computer*, vol. 34, no. 1, pp. 135–137, 2001, doi: 10.1109/2.962984.

[115] Q. Song, Z. Jia, M. Shepperd, S. Ying, and J. Liu, "A general software defect-proneness prediction framework," *IEEE Transactions on Software Engineering*, vol. 37, no. 3, pp. 356–370, 2010, doi: 10.1109/TSE.2010.90.

[116] P. Mohagheghi, R. Conradi, O. M. Killi, and H. Schwarz, "An empirical study of software reuse vs. defect-density and stability," in *International Conference on Software Engineering*, 2004, pp. 282–291, doi: 10.1109/ICSE.2004.1317450.

[117] N. Nagappan and T. Ball, "Use of relative code churn measures to predict system defect density," in *International Conference on Software Engineering*, 2005, pp. 284–292, doi: 10.1145/1062455.1062514.

[118] S. M. A. Shah, M. Morisio, and M. Torchiano, "An overview of software defect density: a scoping study," in *Asia-Pacific Software Engineering Conference*, vol. 1, 2012, pp. 406–415, doi: 10.1109/APSEC.2012.93.

[119] R. Mijumbi, K. Okumoto, A. Asthana, and J. Meekel, "Recent advances in software reliability assurance," in *IEEE International Symposium on Software Reliability Engineering Workshops*, 2018, pp. 77–82, doi: 10.1109/ISSREW.2018.00-27.

[120] T. M. Fehlmann and E. Kranich, "Measuring defects in a six sigma context," in *International Conference on Lean Six Sigma*, 2014.

[121] J. Li, N. B. Moe, and T. Dybå, "Transition from a plan-driven process to scrum: a longitudinal case study on software quality," in *ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, 2010, doi: 10.1145/1852786.1852804.

[122] J. S. Collofello, Z. Yang, J. D. Tvedt, D. Merrill, and I. Rus, "Modeling software testing processes," in *IEEE Fifteenth Annual International Phoenix Conference on Computers and Communications*, 1996, pp. 289–293, doi: 10.1109/PCCC.1996.493647.

[123] P. M. Jacob and M. Prasanna, "A comparative analysis on black box testing strategies," in *International Conference on Information Science*, 2016, pp. 1–6, doi: 10.1109/INFOSCI.2016.7845290.

[124] M. S. Rawat and S. K. Dubey, "Software defect prediction models for quality improvement: a literature study," *International Journal of Computer Science Issues*, vol. 9, no. 5, pp. 288–296, 2012.

[125] V. Vashisht, M. Lal, and G. S. Sureshchandar, "A framework for software defect prediction using neural networks," *Journal of Software Engineering and Applications*, vol. 8, no. 8, pp. 384–394, 2015, doi: 10.4236/jsea.2015.88038.

[126] M. G. Institute, "Prioritizing health: A prescription for prosperity | mckinsey," 2020.

[127] C. I. Agency, "Country Comparisons – Life expectancy at birth," 2022.

[128] CDC, "Chronic Diseases in America | CDC," https://www.cdc.gov, Jan. 2021.

[129] ——, "Health and Economic Costs of Chronic Diseases | CDC," https://www.cdc.gov, Jun. 2021.

[130] Academy of Nutrition and Dietetics, "Disorders of Lipid Metabolism - EXECUTIVE SUMMARY OF RECOMMENDATIONS (2011)," Tech. Rep., 2011.

[131] National Institute of Health, "Nutrient Recommendations : Dietary Reference Intakes (DRI)," Tech. Rep., 2019.

[132] W. C. Willett, J. P. Koplan, R. Nugent, C. Dusenbury, P. Puska, and T. A. Gaziano, "Prevention of Chronic Disease by Means of Diet and Lifestyle Changes," in *Disease Control Priorities in Developing Countries*, 2nd ed., D. T. Jamison, J. G. Breman, A. R. Measham, G. Alleyne, M. Claeson, D. B. Evans, P. Jha, A. Mills, and P. Musgrove, Eds. Washington (DC): World Bank, 2006.

[133] Z. Gervis, "Most People Think Food Labels Are Misleading," Jun. 2018.

[134] A. Persoskie, E. Hennessy, and W. L. Nelson, "US Consumers' Understanding of Nutrition Labels in 2013: The Importance of Health Literacy," *Preventing Chronic Disease*, vol. 14, p. 170066, Sep. 2017, doi: 10.5888/pcd14.170066.

[135] American Heart Association, "Food Labeling Survey," 2019.

[136] I. A. Iles, X. Nan, and L. Verrill, "Nutrient Content Claims: How They Impact Perceived Healthfulness of Fortified Snack Foods and the Moderating Effects of Nutrition Facts Labels," *Health Communication*, vol. 33, no. 10, pp. 1308–1316, Oct. 2018, doi: 10.1080/10410236.2017.1351277.

[137] L. Verrill and C. J. Choinière, "Are Food Allergen Advisory Statements Really Warnings? Variation in Consumer Preferences and Consumption Decisions," *Journal of Food Products Marketing*, vol. 15, no. 2, pp. 139–151, Mar. 2009, doi: 10.1080/10454440802316800.

[138] L. Verrill, D. Wood, S. Cates, A. Lando, and Y. Zhang, "Vitamin-Fortified Snack Food May Lead Consumers to Make Poor Dietary Decisions," *Journal of the Academy of Nutrition and Dietetics*, vol. 117, no. 3, pp. 376–385, Mar. 2017, doi: 10.1016/j.jand.2016.10.008.

[139] J. Labiner-Wolfe, C.-T. J. Lin, and L. Verrill, "Effect of Low-carbohydrate Claims on Consumer Perceptions about Food Products' Healthfulness and Helpfulness for Weight Management," *Journal of Nutrition Education and Behavior*, vol. 42, no. 5, pp. 315–320, Sep. 2010, doi: 10.1016/j.jneb.2009.08.002.

[140] T. D. Giardina, J. Baldwin, D. T. Nystrom, D. F. Sittig, and H. Singh, "Patient perceptions of receiving test results via online portals: A mixed-methods study," *Journal of the American Medical Informatics Association*, vol. 25, no. 4, pp. 440–446, Apr. 2018, doi: 10.1093/jamia/ocx140.

[141] Z. Zhang, Y. Lu, Y. Kou, D. T. Y. Wu, J. Huh-Yoo, and Z. He, "coverUnderstanding Patient Information Needs About Their Clinical Laboratory Results: A Study of Social Q&A Site," *Studies in Health Technology and Informatics*, vol. 264, pp. 1403–1407, 2019, doi: 10.3233/SHTI190458.

[142] H. Kalantarian, N. Alshurafa, and M. Sarrafzadeh, "A Survey of Diet Monitoring Technology," *IEEE Pervasive Computing*, vol. 16, no. 1, pp. 57–65, Jan. 2017, doi: 10.1109/MPRV.2017.1.

[143] D. S. Ludwig, "Technology, Diet, and the Burden of Chronic Disease," *JAMA*, vol. 305, no. 13, pp. 1352–1353, Apr. 2011, doi: 10.1001/jama.2011.380.

[144] K. Poutanen, N. Sozer, and G. Della Valle, "How can technology help to deliver more of grain in cereal foods for a healthy diet?" *Journal of Cereal Science*, vol. 59, no. 3, pp. 327–336, May 2014, doi: 10.1016/j.jcs.2014.01.009.

[145] B. Spring, K. Schneider, H. G. McFadden, J. Vaughn, A. T. Kozak, M. Smith, A. C. Moller, L. H. Epstein, A. DeMott, D. Hedeker, J. Siddique, and D. M. Lloyd-Jones, "Multiple Be-

havior Changes in Diet and Activity: A Randomized Controlled Trial Using Mobile Technology," *Archives of Internal Medicine*, vol. 172, no. 10, pp. 789–796, May 2012, doi: 10.1001/archinternmed.2012.1044.

[146] A. Afshin, D. Babalola, M. Mclean, Z. Yu, W. Ma, C.-Y. Chen, M. Arabi, and D. Mozaffarian, "Information Technology and Lifestyle: A Systematic Evaluation of Internet and Mobile Interventions for Improving Diet, Physical Activity, Obesity, Tobacco, and Alcohol Use," *Journal of the American Heart Association: Cardiovascular and Cerebrovascular Disease*, vol. 5, no. 9, p. e003058, Aug. 2016, doi: 10.1161/JAHA.115.003058.

[147] M. Takemoto, T. M. Manini, D. E. Rosenberg, A. Lazar, Z. Z. Zlatar, S. K. Das, and J. Kerr, "Diet and Activity Assessments and Interventions Using Technology in Older Adults," *American Journal of Preventive Medicine*, vol. 55, no. 4, pp. e105–e115, Oct. 2018, doi: 10.1016/j.amepre.2018.06.005.

[148] M. Little, "Nutrition and Technology in the Management of Diet-Related Chronic Diseases: Providing Evidence-Based Nutrition Recommendations in the Collaborative Development of a Nutrition Optimization Tool for Physicians at Grady Health Systems," Georgia State University, Tech. Rep., Sep. 2020.

[149] E. Reis, *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*. Currency, 2011.

[150] S. Blank and R. Dorf, *The Startup Owner's Manual. 1: The Step-by-Step Guide for Building a Great Company*, 1st ed. Pescadero, Calif: K&S Ranch, 2012.

[151] R. V. Anand and M. Dinakaran, "Issues in scrum Agile development principles and practices in software development," *Indian Journal of Science and Technology*, vol. 8, no. 35, pp. 1–5, 2015, doi: 10.17485/ijst/2015/v8i35/79037.

[152] Digital.ai, Feb 2022, url: https://digital.ai/resource-center/analyst-reports/state-of-agile-report/.

[153] M. Pittman, "Lessons learned in managing object-oriented development," *IEEE Software*, vol. 10, no. 1, pp. 43–53, 1993.

[154] G. Booch, *Object solutions: managing the object-oriented project*. Addison Wesley Longman Publishing Co., Inc., 1995.

[155] I. S. Graham and A. Graham, *Migrating to object technology*. Addison-Wesley Longman Publishing Co., Inc., 1995.

[156] H. Takeuchi and I. Nonaka, "The new new product development game," *Harvard business review*, vol. 64, no. 1, pp. 137–146, 1986.

[157] K. Schwaber and M. Beedle, *Agile software development with scrum. Series in agile software development*. Prentice Hall Upper Saddle River, 2002, vol. 1.

[158] T. Satpathy *et al.*, *A Guide to the Scrum Body of Knowledge (SBOK™ Guide)*, 3rd ed. SCRUMstudy, 2016.

[159] R. Akif and H. Majeed, "Issues and challenges in scrum implementation," *International Journal of Scientific & Engineering Research*, vol. 3, no. 8, pp. 1–4, Aug. 2012.

[160] K. Buffardi, C. Robb, and D. Rahn, "Learning agile with tech startup software engineering projects," in *ACM Conference on Innovation and Technology in Computer Science Education*, Jun. 2017, pp. 28–33, doi: 10.1145/3059009.3059063.

[161] A. Ghezzi and A. Cavallo, "Agile business model innovation in digital entrepreneurship: lean startup approaches," *Journal of Business Research*, vol. 110, pp. 519–537, Mar. 2020, doi: 10.1016/j.jbusres.2018.06.013.

[162] Atlassian, "Jira cloud," 2019, url: https://www.atlassian.com/software/jira.

[163] ClickUp™. url: https://www.clickup.com.

[164] A. Baird and F. Riggins, "Planning and Sprinting: Use of a Hybrid Project Management Methodology within a CIS Capstone Course," *Journal of Information Systems Education*, vol. 23, no. 3, pp. 243–258, Jan. 2012.

[165] S. Ambler and M. Lines, *Choose Your WoW! A Disciplined Agile Approach to Optimizing Your Way of Working (WoW)*, 2nd ed., 2019.

[166] G. Lucassen, F. Dalpiaz, J. M. E. van der Werf, and S. Brinkkemper, "Improving agile requirements: the quality user story framework and tool," *Requirements engineering*, vol. 21, pp. 383–403, 2016.

[167] I. Jacobson, I. Spence, and B. Kerr, "Use-case 2.0," *Communications of the ACM*, vol. 59, pp. 61–69, Apr. 2016, doi: 10.1145/2890778.

[168] I. Jacobson, "Object-oriented development in an industrial environment," *ACM SIGPLAN Notices*, vol. 22, no. 12, pp. 183–191, Dec. 1987, doi: 10.1145/38807.38824.

[169] N. J. Basha, S. A. Moiz, and M. Rizwanullah, "Model Based Software Development: Issues & Challenges," no. 1, p. 6, 2006.

[170] M. Brambilla, J. Cabot, and M. Wimmer, *Model-Driven Software Engineering in Practice*. Springer Cham, 2017, doi: 10.1007/978-3-031-02549-5.

[171] S. Bonnet, J.-L. Voirin, V. Normand, and D. Exertier, "Implementing the MBSE cultural change: organization, coaching and lessons learned," in *INCOSE International Symposium*, vol. 25, no. 1, 2015, pp. 508–523, doi: 10.1002/j.2334-5837.2015.00078.x.

[172] J. Sztipanovits, "Model-Based Software Development," Mar. 2007.

[173] D. Harel and B. Rumpe, "Modeling Languages: Syntax, Semantics and all that Stuff (or, What's the Semantics of "Semantics"?)," Aug. 2004.

[174] H. Stachowiak, *Allgemeine Modelltheorie.* Wien, New York: Springer-Verlag, 1973.

[175] P. Clements, D. Garlan, R. Little, R. Nord, and J. Stafford, "Documenting software architectures: Views and beyond," in *25th International Conference on Software Engineering, 2003. Proceedings.*, May 2003, pp. 740–741, doi: 10.1109/ICSE.2003.1201264.

[176] M. Matinlassi, E. Ovaska, and L. Dobrica, "Quality-Driven Architecture Design and Quality Analysis Method: A Revolutionary Initiation Approach to a Product Line Architecture," Technical Research Centre of Finland, Tech. Rep. VTT Publications 456, 2002.

[177] A. G. Kleppe, J. B. Warmer, and W. Bast, *MDA Explained: The Model Driven Architecture: Practice and Promise*, ser. The Addison-Wesley Object Technology Series. Boston: Addison-Wesley, 2003.

[178] J. S. Topper and N. C. Horner, "Model-Based Systems Engineering in Support of Complex Systems Development," *JOHNS HOPKINS APL TECHNICAL DIGEST*, vol. 32, no. 1, p. 14, 2013.

[179] A. M. Madni and S. Purohit, "Economic Analysis of Model-Based Systems Engineering," *Systems*, vol. 7, no. 1, p. 12, Mar. 2019, doi: 10.3390/systems7010012.

[180] T. E. Bell and T. Thayer, "Software requirements: Are they really a problem?" in *ICSE '76*, 1976.

[181] T. Javed, M. e Maqsood, and Q. S. Durrani, "A study to investigate the impact of requirements instability on software defects," *ACM SIGSOFT Software Engineering Notes*, vol. 29, no. 3, pp. 1–7, May 2004, doi: 10.1145/986710.986727.

[182] SysML Forum, "What Is the Relationship between SysML and UML?" 2018.

[183] K. G. Young, "Defense space application of MBSE-closing the culture chasms," in *AIAA SPACE Conference and Exposition*, 2015, doi: 10.2514/6.2015-4620.

[184] L. Wang, M. Izygon, S. Okon, H. Wagner, and L. Garner, "Effort to accelerate MBSE adoption and usage at JSC," in *AIAA SPACE*, 2016, doi: 10.2514/6.2016-5542.

[185] S. Friedenthal, R. Griego, and M. Sampson, "Incose model based systems engineering (mbse) initiative," in *INCOSE 2007 symposium*, vol. 11, 2007.

[186] Sparx Systems. Enterprise architect 15.2 user guide. url: https://sparxsystems.com/enterprise_architect_user_guide/15.2.

[187] ISO/IEC/IEEE, "Systems and software engineering – architecture description," *ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000)*, pp. 1 –46, Jan. 2011, doi: 10.1109/IEEESTD.2011.6129467.

[188] D. D. Walden, G. J. Roedler, and K. Forsberg, "INCOSE systems engineering handbook version 4: updating the reference for practitioners," in *INCOSE International Symposium*, vol. 25, no. 1, 2015, pp. 678–686, doi: 10.1002/j.2334-5837.2015.00089.x.

[189] J. A. Estefan, "Survey of model-based systems engineering (MBSE) methodologies," INCOSE MBSE Initiative, Tech. Rep., 2008.

[190] P. Zimmerman, "A review of model-based systems engineering practices and recommendations for future directions in the department of defense," in *2nd Systems Engineering in the Washington Metropolitan Area (SEDC 2014) Conference, Chantilly, VA*, 2014.

[191] R. A. Noguchi, "A roadmap for advancing the state of the practice of model based systems engineering for government acquisition," in *INCOSE International Symposium*, vol. 29, no. 1, 2019, pp. 678–690, doi: 10.1002/j.2334-5837.2019.00628.x.

[192] K. Henderson and A. Salado, "Value and benefits of model-based systems engineering (MBSE): evidence from the literature," *Systems Engineering*, vol. 24, no. 1, pp. 51–66, 2021, doi: 10.1002/sys.21566.

[193] A. Mersino, "What is Agile and Why is it Important?" Nov. 2019.

[194] J. Cho, "ISSUES AND CHALLENGES OF AGILE SOFTWARE DEVELOPMENT WITH SCRUM," *Issues In Information Systems*, 2008, doi: 10.48009/2_iis_2008_188-195.

[195] G. Lozo, "A Flexible Hybrid Method for IT Project Management," 2012.

[196] M. Špundak, "Mixed Agile/Traditional Project Management Methodology – Reality or Illusion?" *Procedia - Social and Behavioral Sciences*, vol. 119, pp. 939–948, Mar. 2014, doi: 10.1016/j.sbspro.2014.03.105.

[197] J. Binder, L. I. Aillaud, and L. Schilli, "The Project Management Cocktail Model: An Approach for Balancing Agile and ISO 21500," *Procedia - Social and Behavioral Sciences*, vol. 119, pp. 182–191, Mar. 2014, doi: 10.1016/j.sbspro.2014.03.022.

[198] I. Hadar, S. Sherman, E. Hadar, and J. J. Harrison, "Less is more: Architecture documentation for agile development," in *International Workshop on Cooperative and Human Aspects of Software Engineering*, 2013, pp. 121–124, doi: 10.1109/CHASE.2013.6614746.

[199] C. J. Stettina and W. Heijstek, "Necessary and neglected? an empirical study of internal documentation in agile software development teams," in *ACM International Conference on Design of Communication*, 2011, pp. 159–166, doi: 10.1145/2038476.2038509.

[200] J. Pasuksmit, P. Thongtanunam, and S. Karunasekera, "Towards just-enough documentation for agile effort estimation: what information should be documented?" in *IEEE International Conference on Software Maintenance and Evolution*, 2021, pp. 114–125, doi: 10.1109/ICSME52107.2021.00017.

[201] C. R. Prause and Z. Durdik, "Architectural design and documentation: Waste in agile development?" in *International Conference on Software and System Process*, 2012, pp. 130–134, doi: 10.1109/ICSSP.2012.6225956.

[202] E. Rubin and H. Rubin, "Supporting agile software development through active documentation," *Requirements Engineering*, vol. 16, pp. 117–132, 2011, doi: 10.1007/s00766-010-0113-9.

[203] B. Selic, "Agile documentation, anyone?" *IEEE Software*, vol. 26, no. 6, pp. 11–12, 2009, doi: 10.1109/MS.2009.167.

[204] S. Balaji and M. S. Murugaiyan, "Waterfall vs. V-Model vs. Agile: A comparative study on SDLC," *International Journal of Information Technology and Business Management*, vol. 2, no. 1, pp. 26–30, 2012.

[205] P. J. Younse, J. E. Cameron, and T. H. Bradley, "Comparative analysis of a model-based systems engineering approach to a traditional systems engineering approach for architecting a robotic space system through knowledge categorization," *Systems Engineering*, vol. 24, no. 3, pp. 177–199, 2021, doi: 10.1002/sys.21573.

[206] I. Graessler, J. Hentze, and T. Bruckmann, "V-MODELS FOR INTERDISCIPLINARY SYSTEMS ENGINEERING," in *DS 92: Proceedings of the DESIGN 2018 15th International Design Conference*, 2018, pp. 747–756, doi: 10.21278/idc.2018.0333.

[207] B. Liu, H. Zhang, and S. Zhu, "An Incremental V-Model Process for Automotive Development," in *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*, Dec. 2016, pp. 225–232, doi: 10.1109/APSEC.2016.040.

[208] R. Sell and M. Tamre, "Integration of V-model and SysML for advanced mechatronics system design," in *The 6th International Workshop on Research and Education in Mechatronics REM*, 2005, pp. 276–280.

[209] R. Dove and B. Schindel, "Agility in Systems Engineering – Findings from Recent Studies," Tech. Rep. 14-April-2018, Apr. 2018.

[210] R. Turner, "Toward Agile systems engineering processes," *Crosstalk. The Journal of Defense Software Engineering*, pp. 11–15, 2007.

[211] R. Haberfellner and O. de Weck, "Agile SYSTEMS ENGINEERING versus AGILE SYSTEMS engineering," *INCOSE International Symposium*, vol. 15, no. 1, pp. 1449–1465, 2005, doi: 10.1002/j.2334-5837.2005.tb00762.x.

[212] E. Stelzmann, "Contextualizing agile systems engineering," *IEEE Aerospace and Electronic Systems Magazine*, vol. 27, no. 5, pp. 17–22, May 2012, doi: 10.1109/MAES.2012.6226690.

[213] I. Signoretti, L. Salerno, S. Marczak, and R. Bastos, "Combining User-Centered Design and Lean Startup with Agile Software Development: A Case Study of Two Agile Teams," in *Agile Processes in Software Engineering and Extreme Programming*, ser. Lecture Notes in Business Information Processing, V. Stray, R. Hoda, M. Paasivaara, and P. Kruchten, Eds. Cham: Springer International Publishing, 2020, pp. 39–55, doi: 10.1007/978-3-030-49392-9_3.

[214] K. Buffardi, C. Robb, and D. Rahn, "Learning agile with tech startup software engineering projects," in *ACM Conference on Innovation and Technology in Computer Science Education*, Jun. 2017, pp. 28–33, doi: 10.1145/3059009.3059063.

[215] A. J. Lattanze, *Architecting Software Intensive Systems: A Practitioners Guide*. Auerbach Publications, 2008.

[216] M. Saunders, P. Lewis, and A. Thornhill, *Research Methods*, 8th ed. Pearson Education, 2019.

[217] U. Sekaran and R. Bougie, *Research methods for business: A skill building approach*. john wiley & sons, 2016.

[218] R. K. Yin *et al.*, "Design and methods," *Case study research*, vol. 3, no. 9.2, 2003.

[219] L. K. Soiferman, "Compare and contrast inductive and deductive research approaches." *Online Submission*, 2010.

[220] C. Williams *et al.*, "Research methods," *Journal of Business & Economics Research (JBER)*, vol. 5, no. 3, 2007.

[221] M. Dixon-Woods, S. Agarwal, D. Jones, B. Young, and A. Sutton, "Synthesising qualitative and quantitative evidence: a review of possible methods," *Journal of health services research & policy*, vol. 10, no. 1, pp. 45–53, 2005.

[222] J. Brannen, "Combining qualitative and quantitative approaches: an overview," *Mixing methods: Qualitative and quantitative research*, pp. 3–37, 2017.

[223] T. D. Cook and C. S. Reichardt, *Qualitative and Quantitative Methods in Evaluation Research*. Sage Publications, 1979.

[224] A. Tashakkori, C. Teddlie, and C. B. Teddlie, *Mixed methodology: Combining qualitative and quantitative approaches*. sage, 1998, vol. 46.

[225] D. Buchanan, D. Boddy, and J. McCalman, "Getting in, getting on, getting out, and getting back," in *Doing Research in Organizations (RLE: Organizations)*. Routledge, 2013, pp. 63–77.

[226] I. Walsh, J. A. Holton, L. Bailyn, W. Fernandez, N. Levina, and B. Glaser, "What grounded theory is... a critically reflective conversation among scholars," *Organizational Research Methods*, vol. 18, no. 4, pp. 581–599, 2015.

[227] ——, "Rejoinder: Moving the management field forward," *Organizational Research Methods*, vol. 18, no. 4, pp. 620–628, 2015.

[228] Merriam-Webster, "strategy," url: https://www.merriam-webster.com/dictionary/strategy.

[229] M. Battiste, N. Chomsky, N. K. Denzin, M. Fine, R. Gill, S. Grande, B. L. Hall, P. Lather, Z. Leonardo, Y. S. Lincoln *et al.*, *Dissident knowledge in higher education*. University of Regina Press, 2018.

[230] F. Lau and C. Kuziemsky, *Handbook of eHealth Evaluation: An Evidence-based Approach*, 02 2017.

[231] H. Tang, *Engineering research: Design, methods, and publication*. John Wiley & Sons, 2020.

[232] W. R. Shadish, T. D. Cook, and D. T. Campbell, *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*. Houghton Mifflin Company, 2002.

[233] L. Langbein, *Public program evaluation: A statistical guide*. Routledge, 2014.

[234] S. E. Maxwell, H. D. Delaney, and K. Kelley, *Designing experiments and analyzing data: A model comparison perspective*. Routledge, 2017.

[235] D. T. Campbell and J. C. Stanley, *Experimental and Quasi-Experimental Designs for Research*. Houghton Mifflin Company, 1963.

[236] C. S. Reichardt, *Quasi-Experimentation: A Guide to Design and Analysis*. Guilford Publications, 2019.

[237] D. Schwartz, B. Fischhoff, T. Krishnamurti, and F. Sowell, "The hawthorne effect and energy awareness," *Proceedings of the National Academy of Sciences*, vol. 110, no. 38, pp. 15 242–15 246, 2013, doi: 10.1073/pnas.1301687110.

[238] Slack. url: https://slack.com.

[239] R. Tamrakar and M. Jørgensen, "Does the use of Fibonacci numbers in planning poker affect effort estimates?" in *International Conference on Evaluation & Assessment in Software Engineering*, 2012, doi: 10.1049/ic.2012.0030.

[240] (2022) Powerbi™. url: https://powerbi.microsoft.com/en-us/.

[241] K. R. Das and A. Imon, "A brief review of tests for normality," *American Journal of Theoretical and Applied Statistics*, vol. 5, no. 1, pp. 5–12, 2016, doi: 10.11648/j.ajtas.20160501.12.

[242] A. P. Rovai, J. D. Baker, and M. K. Ponton, *Social Science Research Design and Statistics*. Watertree Press, 2013.

[243] D. Rasch, F. Teuscher, and V. Guiard, "How robust are tests for two independent samples?" *Journal of Statistical Planning and Inference*, vol. 137, no. 8, pp. 2706–2720, 2007, doi: 10.1016/j.jspi.2006.04.011.

[244] (2022) Prism™. url: https://www.graphpad.com/.

[245] P. Raimond, *Management projects: design, research, and presentation*. Chapman & Hall, 1993.

[246] C. R. Rogers, *On becoming a person: A therapist's view of psychotherapy*. Constable London, 1961.

[247] E. C. Matthay and M. M. Glymour, "A graphical catalog of threats to validity: Linking social science with epidemiology," *Epidemiology*, vol. 31, no. 3, pp. 376–384, 2020, doi: 10.1097/EDE.0000000000001161.

[248] D. C. Howell, "Chi-square test: analysis of contingency tables," in *International Encyclopedia of Statistical Science*. Springer, 2011, pp. 250–252, doi: 10.1007/978-3-642-04898-2_174.

[249] A. Field, *Discovering Statistics Using IBM SPSS Statistics*. Sage, 2013.

[250] V. B. Kampenes, T. Dybå, J. E. Hannay, and D. I. Sjøberg, "A systematic review of quasi-experiments in software engineering," *Information and Software Technology*, vol. 51, no. 1, pp. 71–82, 2009, doi: 10.1016/j.infsof.2008.04.006.

[251] M. D. Gall, J. P. Gall, and W. R. Borg, *Educational Research: An Introduction*. Pearson, 2007.

[252] A. P. Field, "Is the meta-analysis of correlation coefficients accurate when population correlations vary?" *Psychological Methods*, vol. 10, no. 4, pp. 444–467, 2005, doi: 10.1037/1082-989X.10.4.444.

[253] J. L. Gastwirth, Y. R. Gel, and W. Miao, "The impact of Levene's test of equality of variances on statistical theory and practice," *Statistical Science*, vol. 24, no. 3, pp. 343–360, 2009, doi: 10.1214/09-STS301.

[254] E. Schmider, M. Ziegler, E. Danay, L. Beyer, and M. Bühner, "Is it really robust?" *Methodology*, vol. 6, no. 4, pp. 147–151, 2010, doi: 10.1027/1614-2241/a000016.

[255] Microsoft 365. url: https://www.office.com.

[256] diagrams.net. Draw.io. url: https://draw.io.

Appendix A

System Model using sMBSAP: Diagram Report

This appendix includes an exported report from the MBSE tool (Sparx EA). This “Diagram Report” provides a summary of the diagrams included within the model. There are a lot of other details captured in the model but are not shown in the “Diagram Report”.

Diagram Report

a. Requirements and Use Cases

Version 1.0 • Final

Date Generated: 2/21/2022

Author: Moe Huss

CREATED WITH ENTERPRISE ARCHITECT

Table of Contents

1.1.1 User Roles diagram

Class diagram in package 'User Roles'

User Roles
Version 1.0
Author: Moe Huss

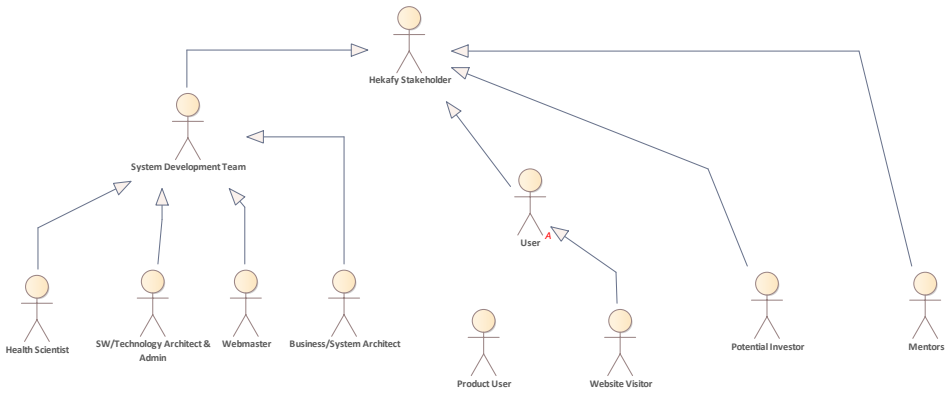


Figure 1: User Roles

1.1.2 Epics, use cases, user stories and requirements diagram

Class diagram in package 'User stories (requirements) and use cases'

Epics, use cases, user stories and requirements
Version 1.0
Author: Moe Huss

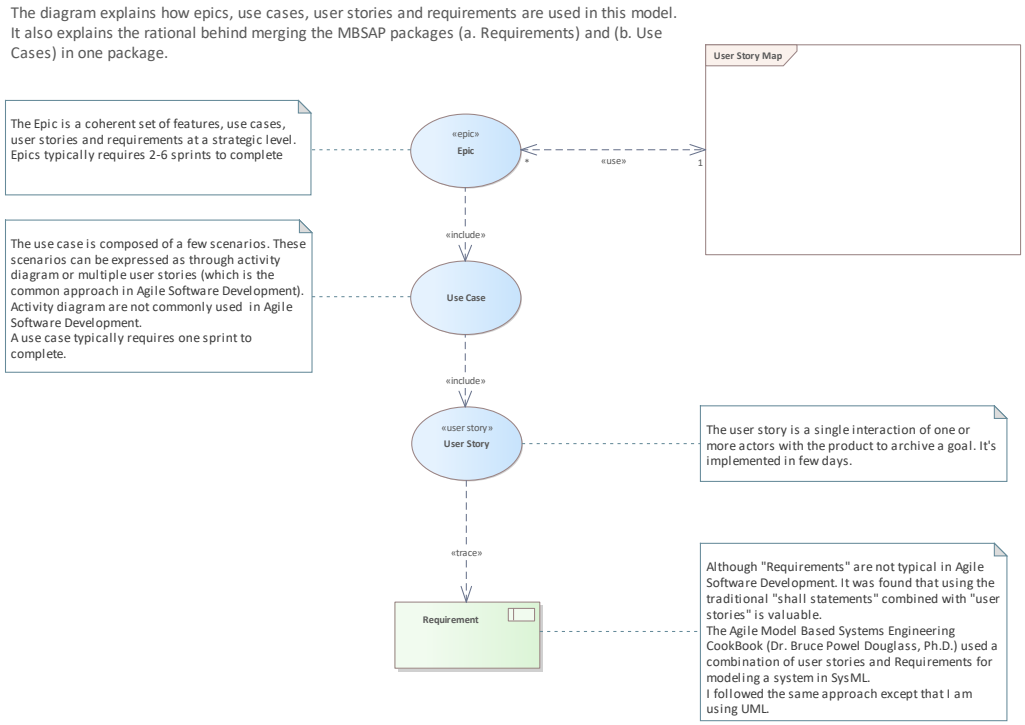


Figure 2: Epics, use cases, user stories and requirements

1.1.3 User Story Map diagram

Logical diagram in package 'User stories (requirements) and use cases'

The user story map enables the team to groom the product backlog and plan the product releases more effectively.
A user story map captures the journey a customer takes with the product including activities and tasks they perform with the system.
The sprints conducted using Agile Scrum vs the sprints conducted using sMBSAP are also shown.

User Story Map
Version 1.0
Author: Moe Huss

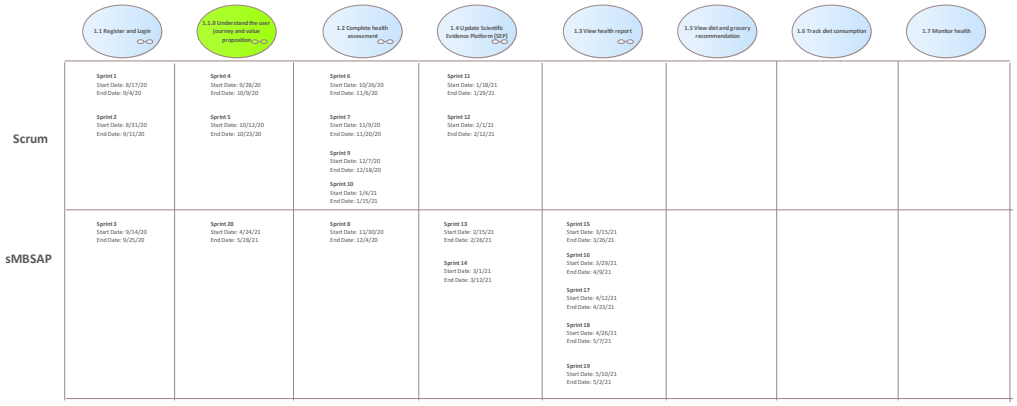


Figure 3: User Story Map

1.1.3.1 System Epics diagram

Use Case diagram in package 'User stories (requirements) and use cases'

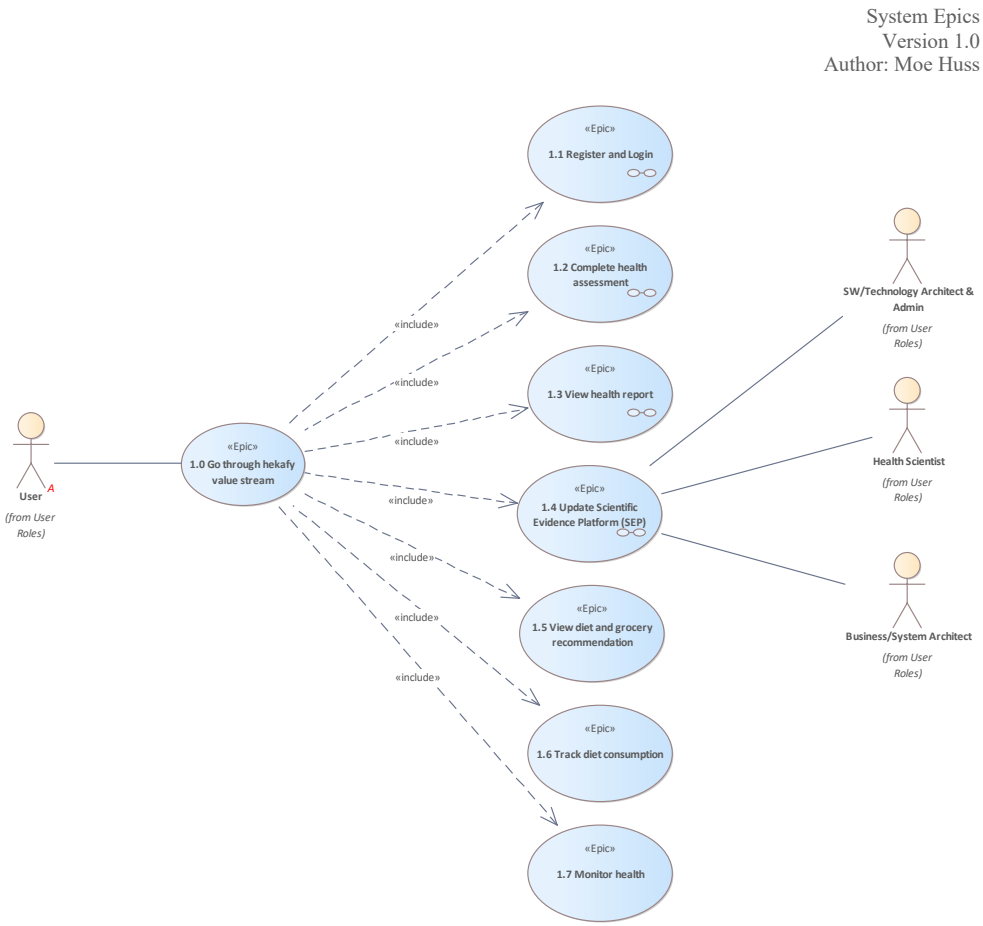


Figure 4: System Epics

1.1.3.2 Register and login use cases diagram

Use Case diagram in package 'User stories (requirements) and use cases'

The home

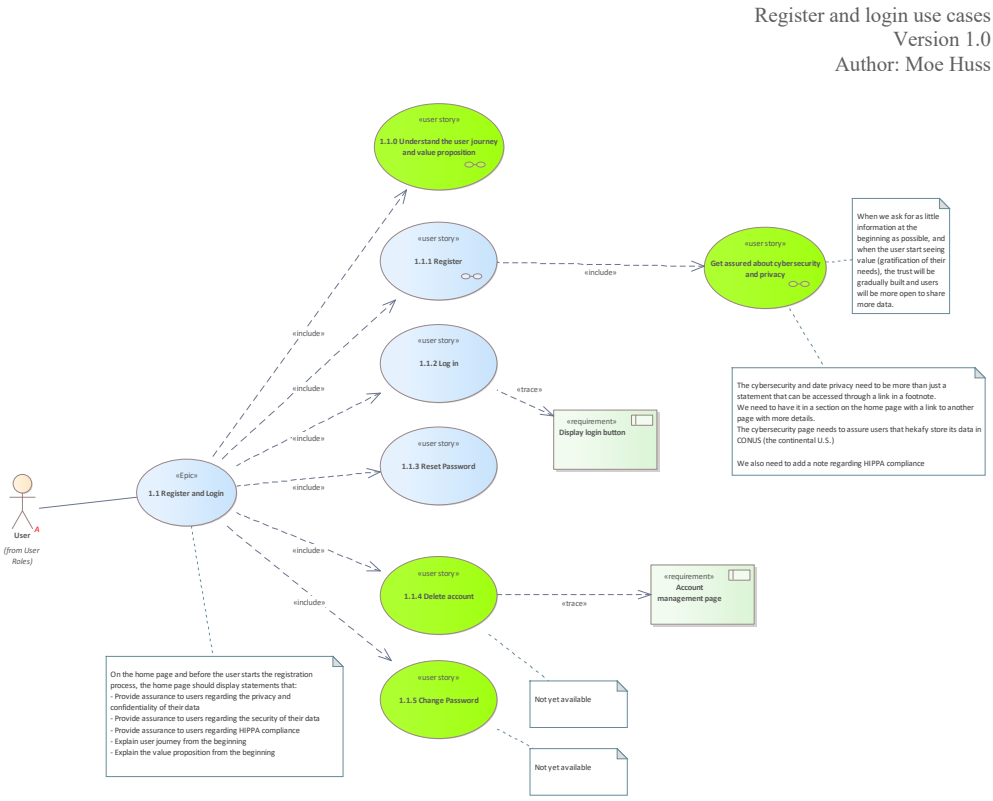


Figure 5: Register and login use cases

1.1.3.2.1 1.1.0 Understand the user journey and value proposition at the beginning diagram

Use Case diagram in package 'User stories (requirements) and use cases'

1.1.0 Understand the user journey and value proposition at the beginning
Version 1.0
Author: Moe Huss

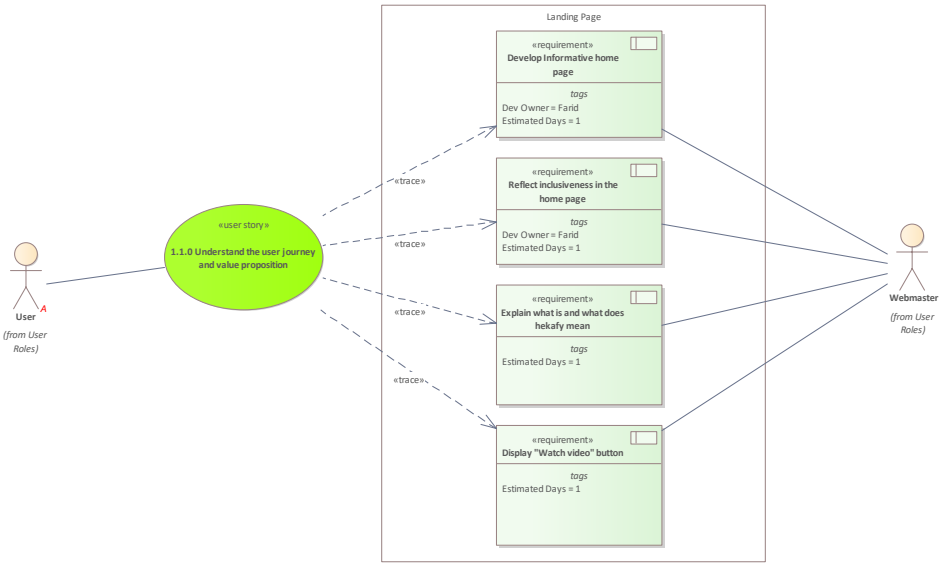


Figure 6: 1.1.0 Understand the user journey and value proposition at the beginning

1.1.3.2.2 Register diagram

Use Case diagram in package 'User stories (requirements) and use cases'

Register
Version 1.0
Author: Moe Huss

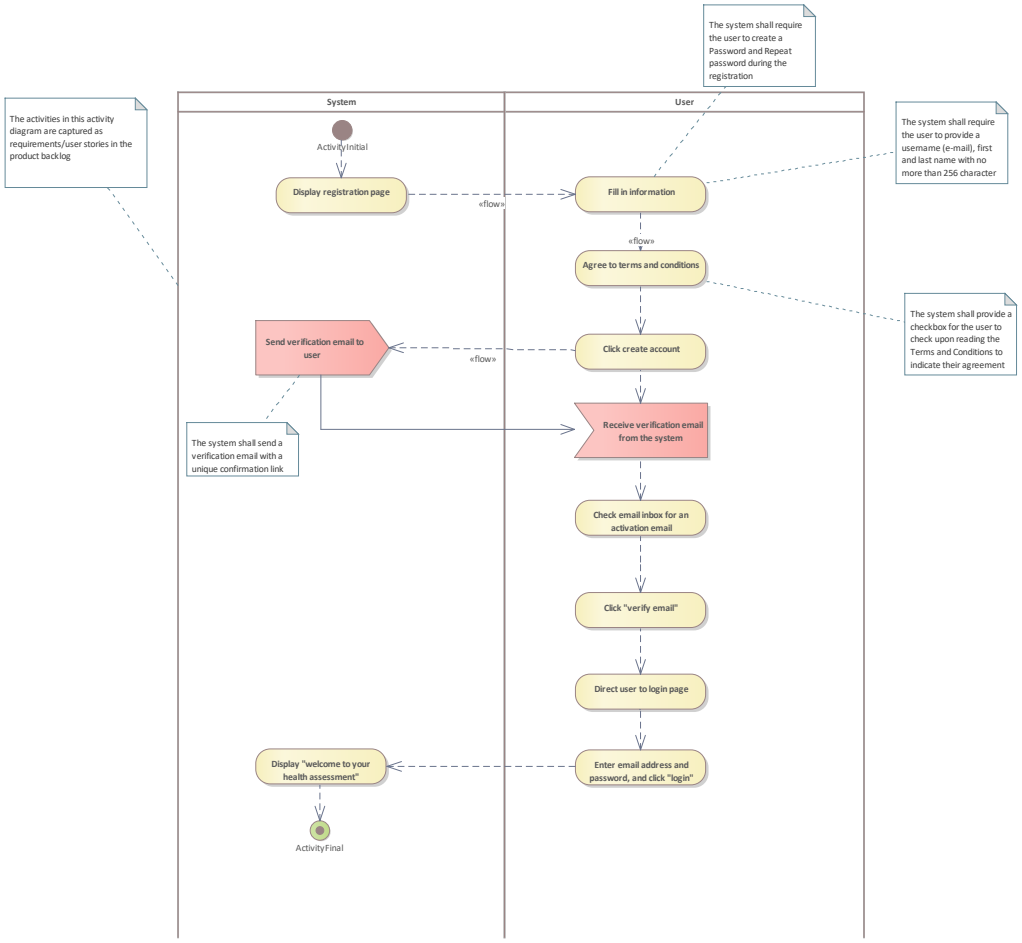


Figure 7: Register

1.1.3.2.3 Get assured about cybersecurity and privacy diagram

Use Case diagram in package 'User stories (requirements) and use cases'

Get assured about cybersecurity and privacy
Version 1.0
Author: Moe Huss

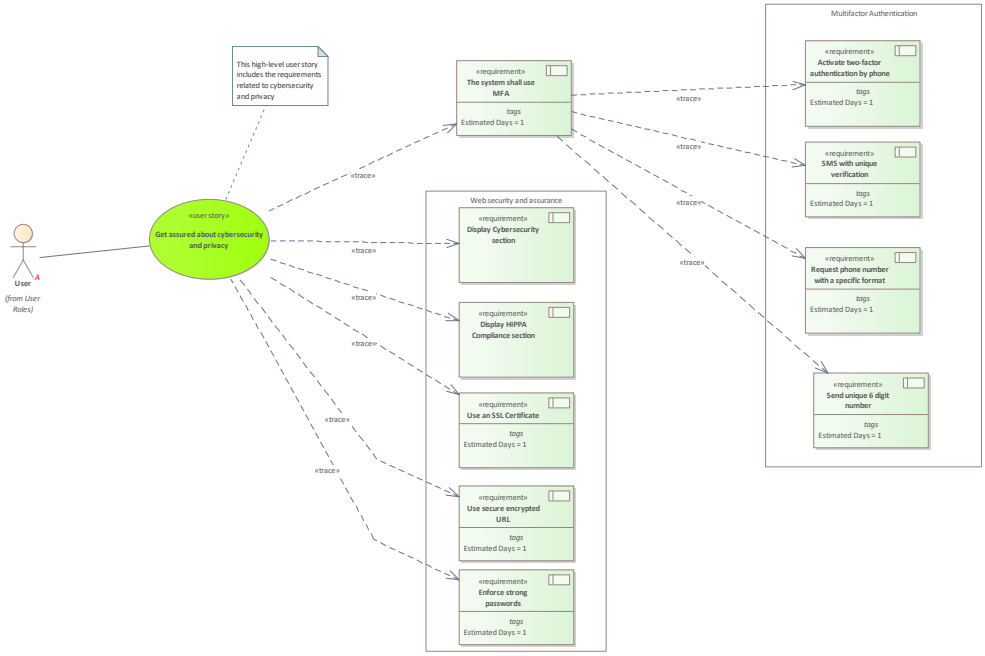


Figure 8: Get assured about cybersecurity and privacy

1.1.3.3 Complete health assessment use cases diagram

Use Case diagram in package 'User stories (requirements) and use cases'

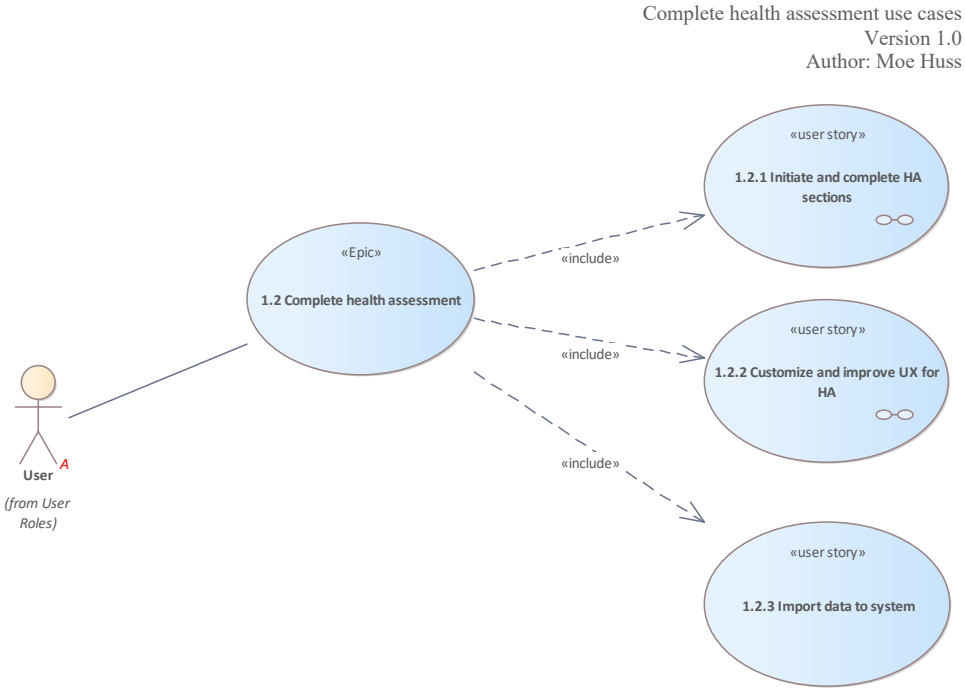


Figure 9: Complete health assessment use cases

1.1.3.4 Initiate and complete HA diagram

Use Case diagram in package 'User stories (requirements) and use cases'

Initiate and complete HA
Version 1.0
Author: Moe Huss

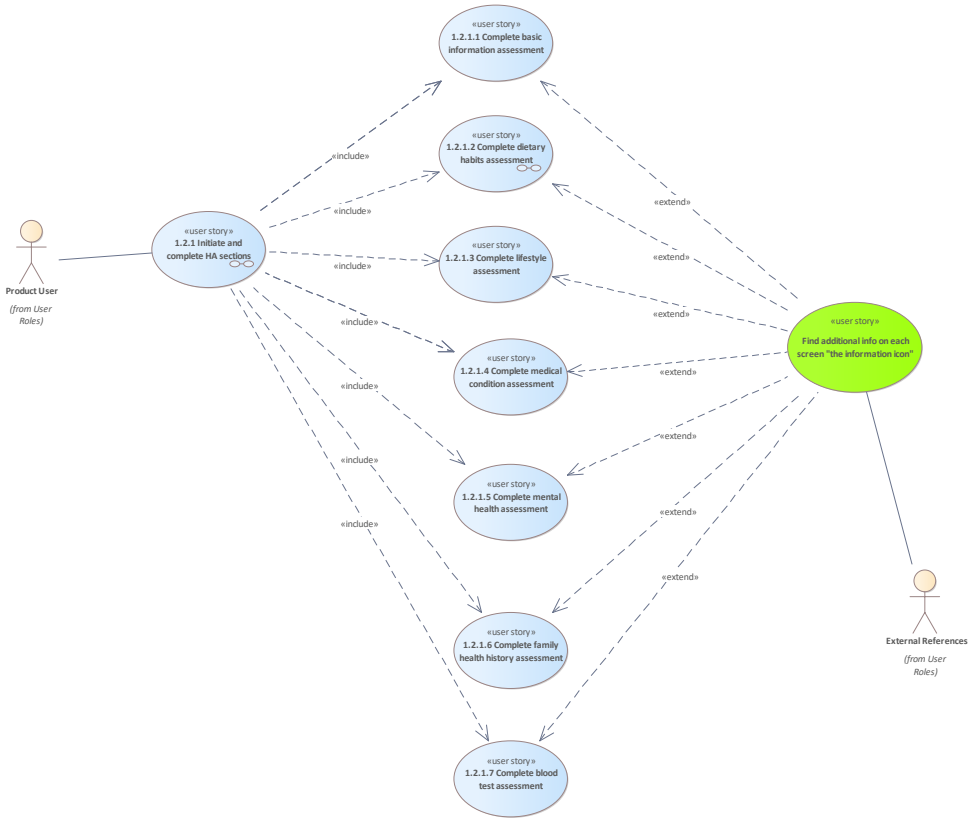


Figure 10: Initiate and complete HA

1.1.3.4.1 Initiate and complete Health Assessment diagram

Use Case diagram in package 'User stories (requirements) and use cases'

Initiate and complete Health Assessment
Version 1.0
Author: Moe Huss

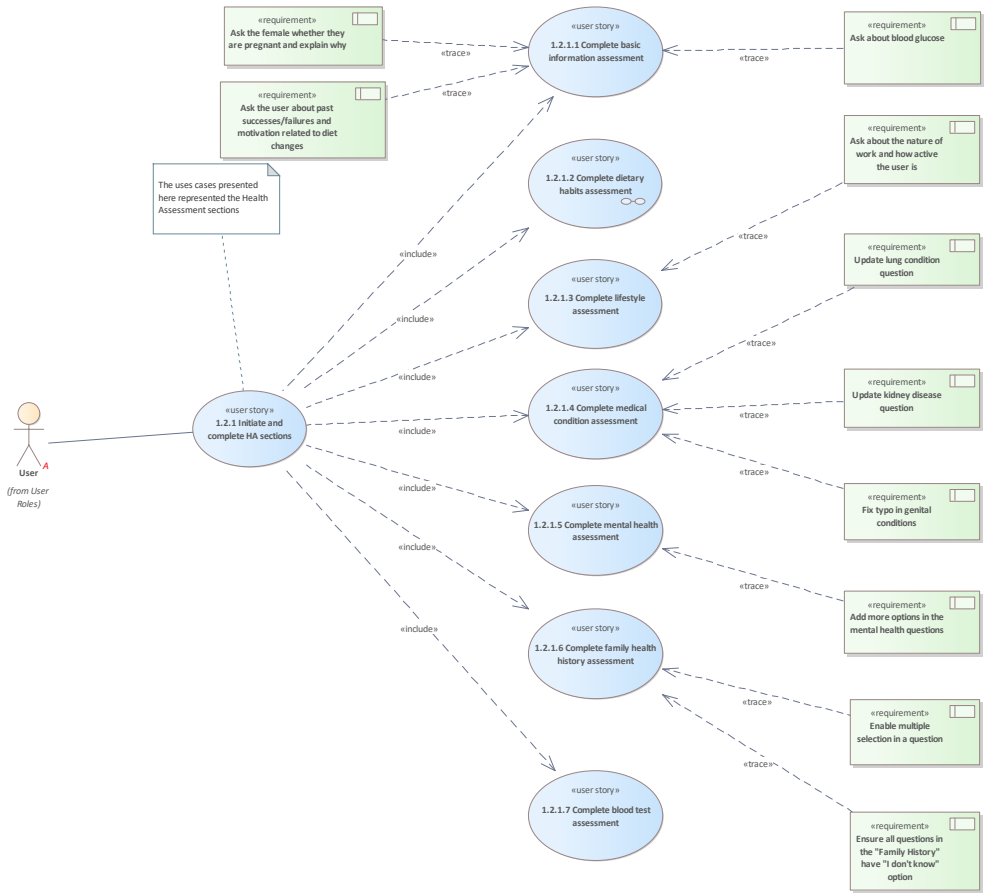


Figure 11: Initiate and complete Health Assessment

1.1.3.4.1.1 Complete dietary habits assessment diagram

Use Case diagram in package 'User stories (requirements) and use cases'

Complete dietary habits assessment
Version 1.0
Author: Moe Huss

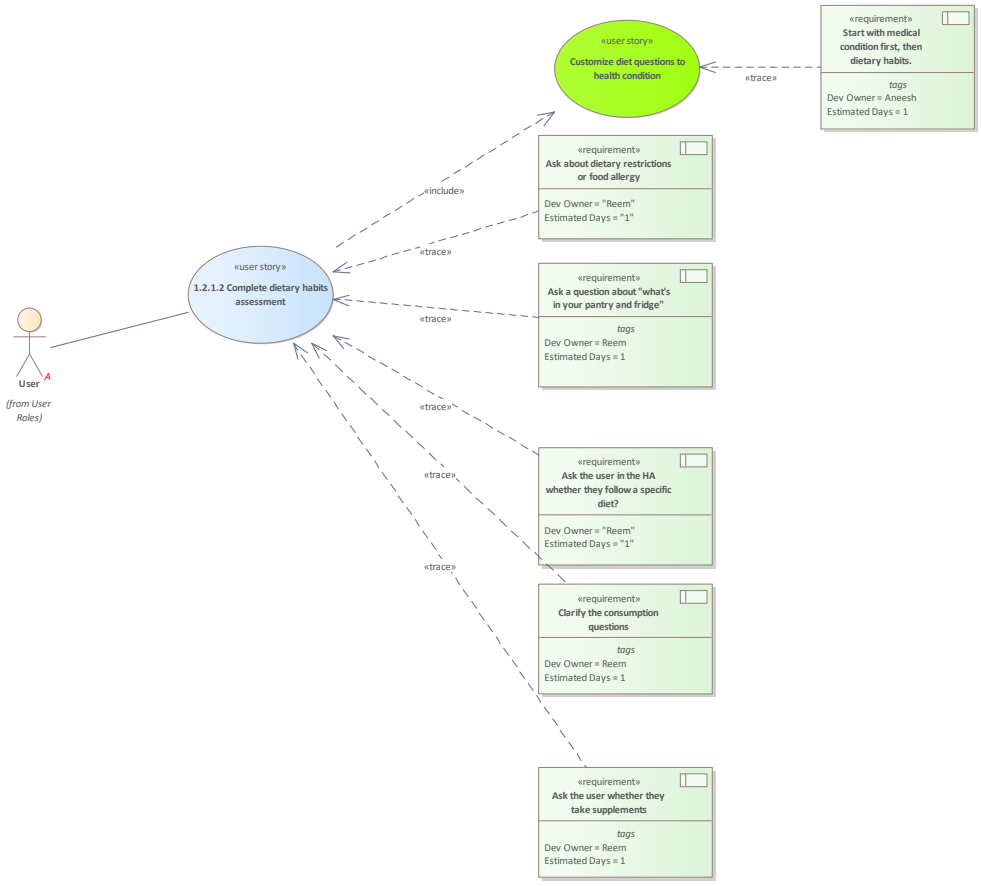


Figure 12: Complete dietary habits assessment

1.1.3.4.2 Customize and improve UX for HA diagram

Use Case diagram in package 'User stories (requirements) and use cases'

Customize and improve UX for HA
Version 1.0
Author: Moe Huss

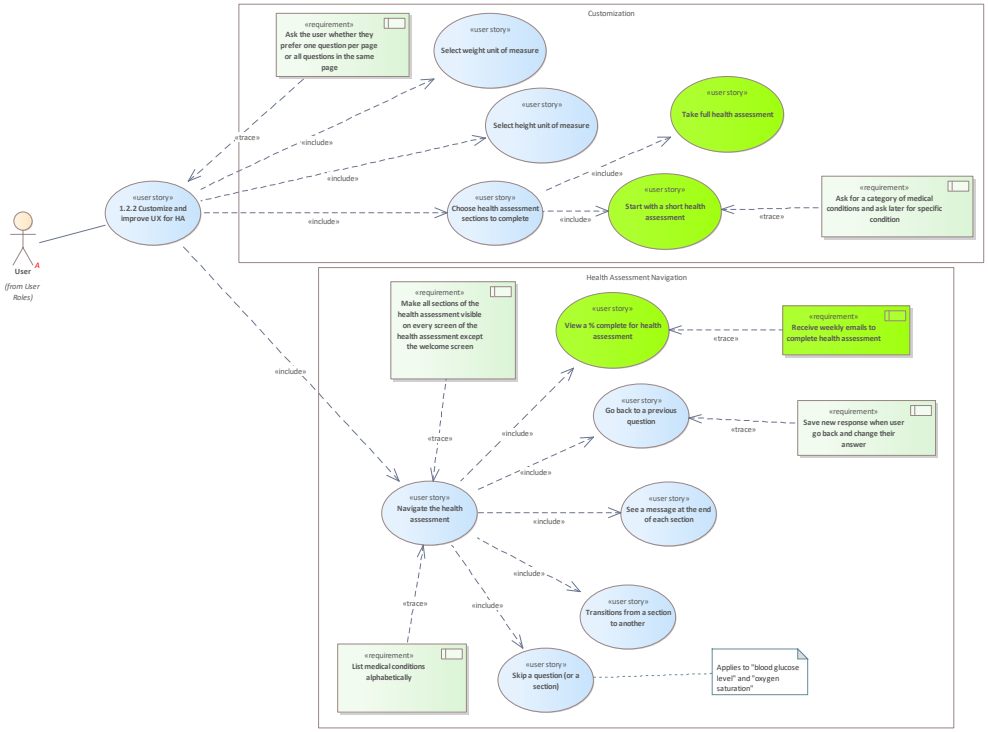


Figure 13: Customize and improve UX for HA

1.1.3.4.3 Import Data to System diagram

Use Case diagram in package 'User stories (requirements) and use cases'

Import Data to System
Version 1.0
Author: Moe Huss

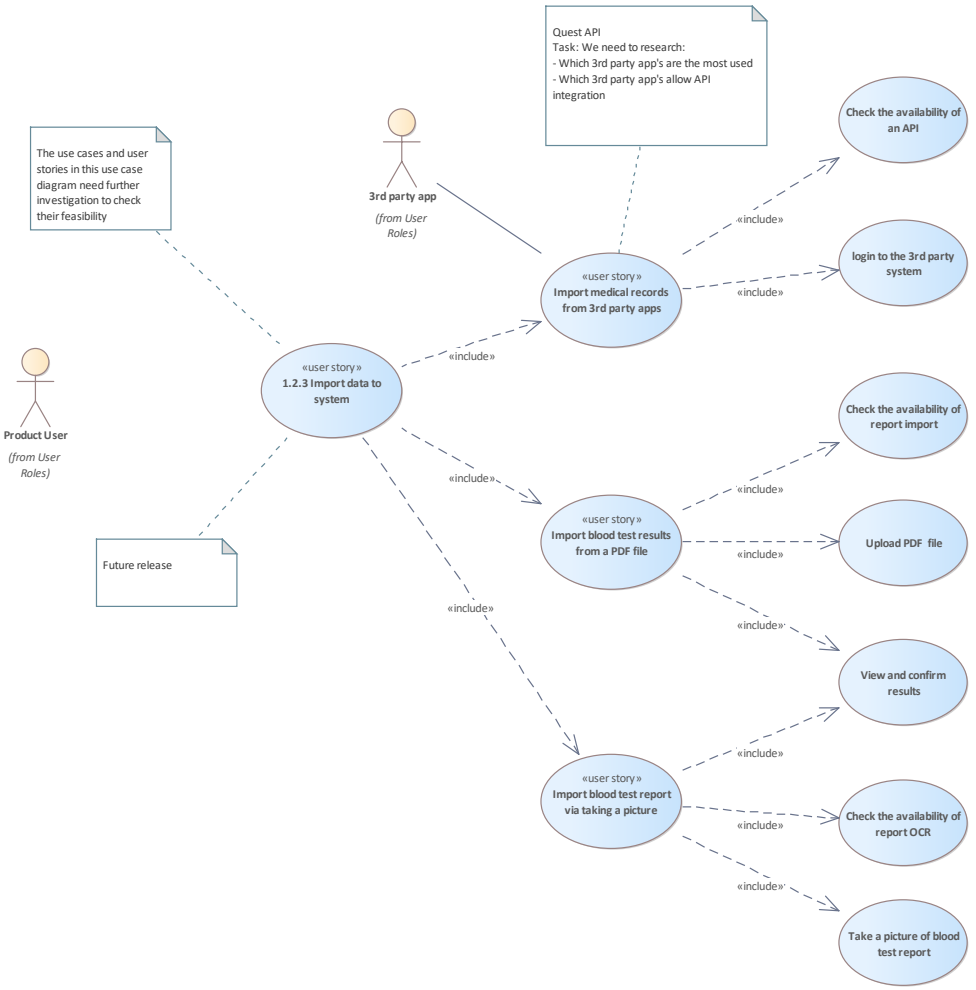


Figure 14: Import Data to System

1.1.3.5 Health Rporter diagram

Use Case diagram in package 'User stories (requirements) and use cases'

Health Rporter
Version 1.0
Author: Moe Huss

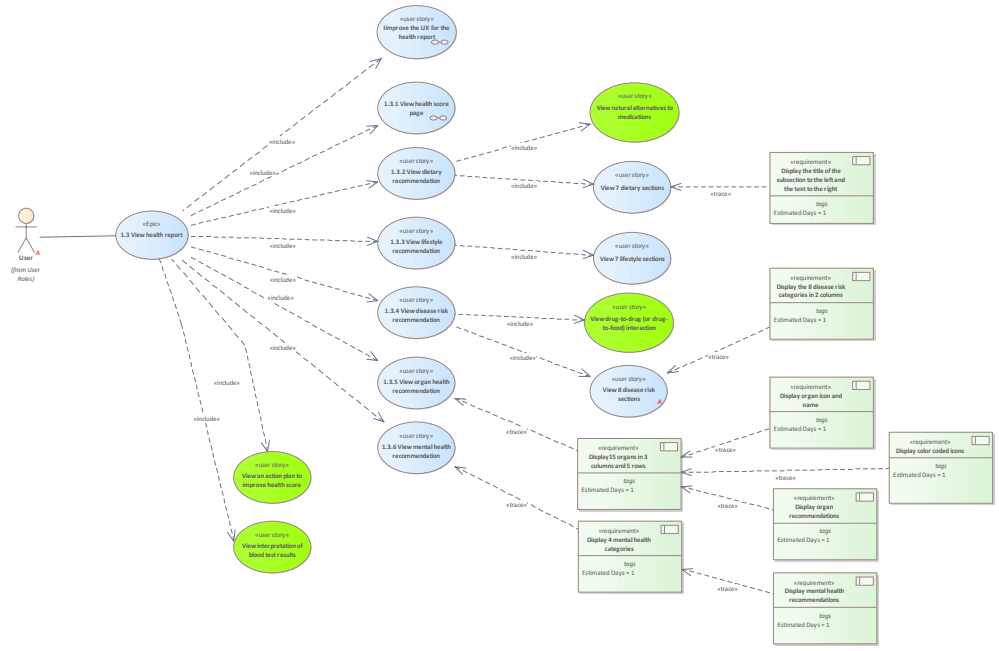


Figure 15: Health Rporter

1.1.3.6 View health report diagram

Use Case diagram in package 'User stories (requirements) and use cases'

View health report
Version 1.0
Author: Moe Huss

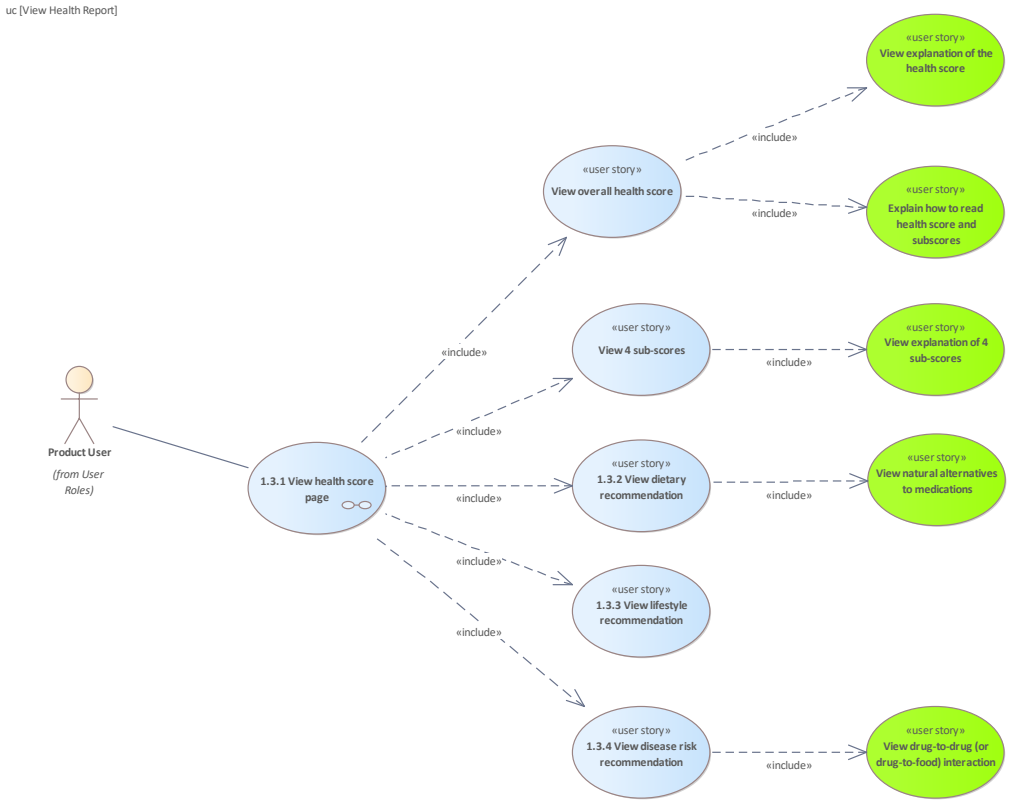


Figure 16: View health report

1.1.3.6.1 1.2.2 Customize and improve UX for HA diagram

Use Case diagram in package 'User stories (requirements) and use cases'

1.2.2 Customize and improve UX for HA
Version 1.0
Author: Moe Huss

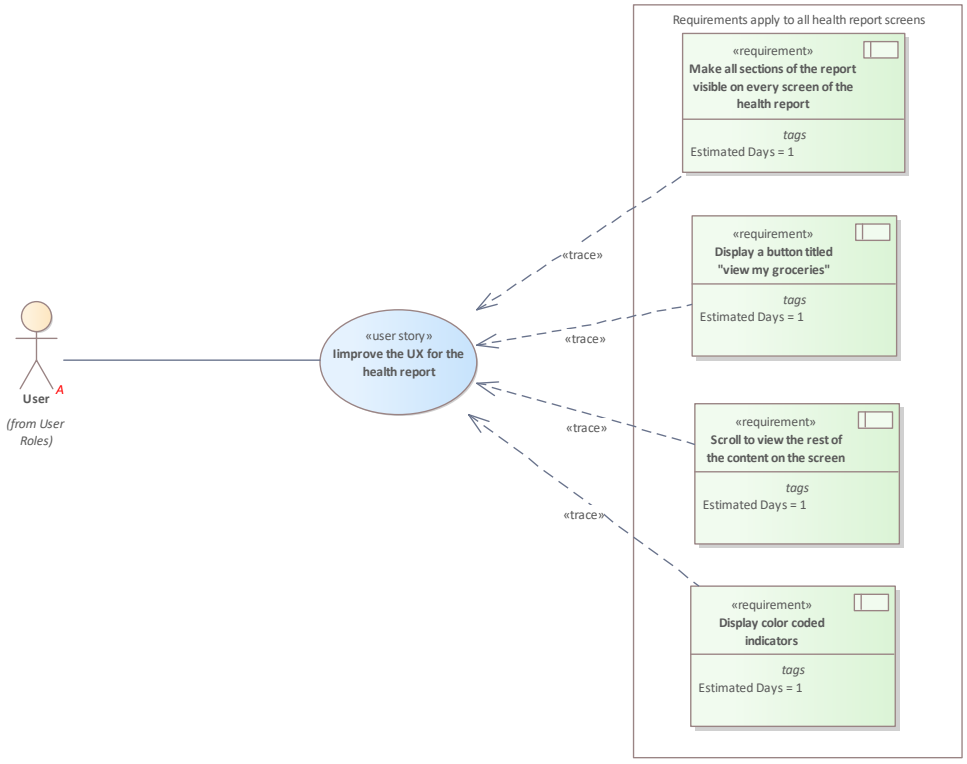


Figure 17: 1.2.2 Customize and improve UX for HA

1.1.3.6.2 View health score page diagram

Use Case diagram in package 'User stories (requirements) and use cases'

View health score page
Version 1.0
Author: Moe Huss

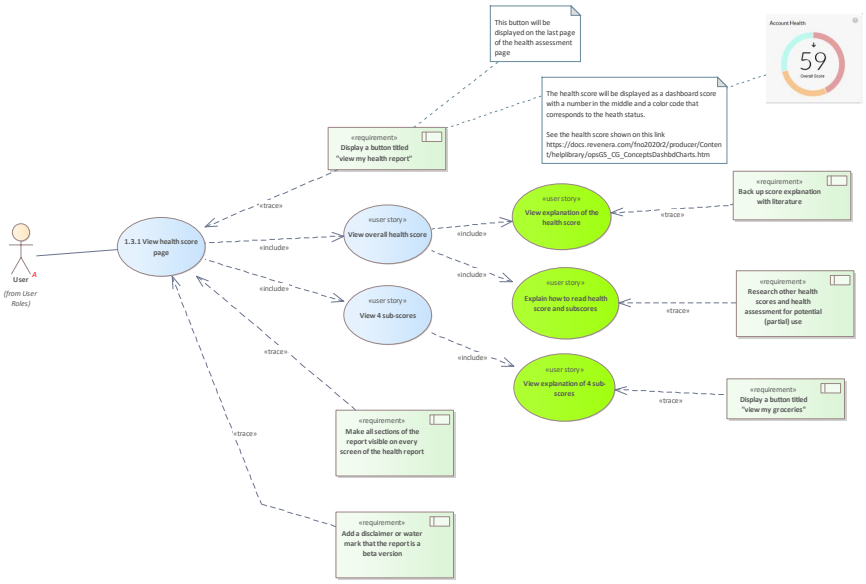


Figure 18: View health score page

1.1.3.7 1.4 Update Scientific Evidence Platform (SEP) diagram

Use Case diagram in package 'User stories (requirements) and use cases'

1.4 Update Scientific Evidence Platform (SEP)
Version 1.0
Author: Moe Huss

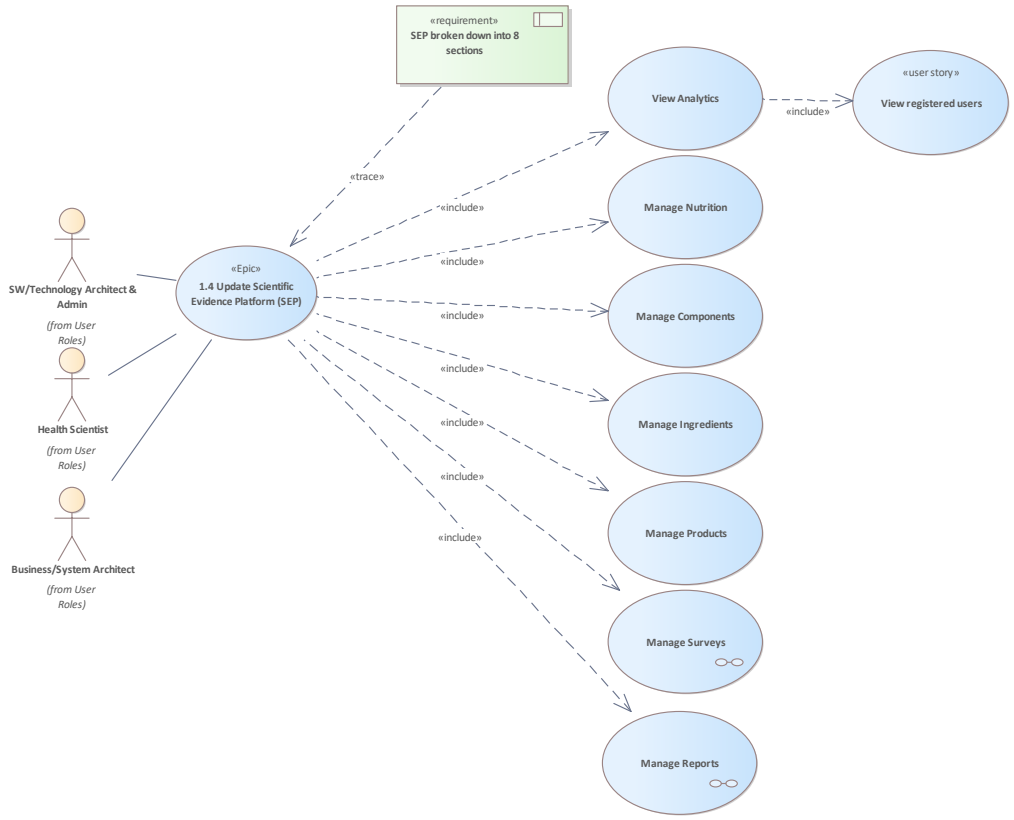


Figure 19: 1.4 Update Scientific Evidence Platform (SEP)

1.1.3.7.1 Manage Reports diagram

Use Case diagram in package 'User stories (requirements) and use cases'

Manage Reports
Version 1.0
Author: Moe Huss

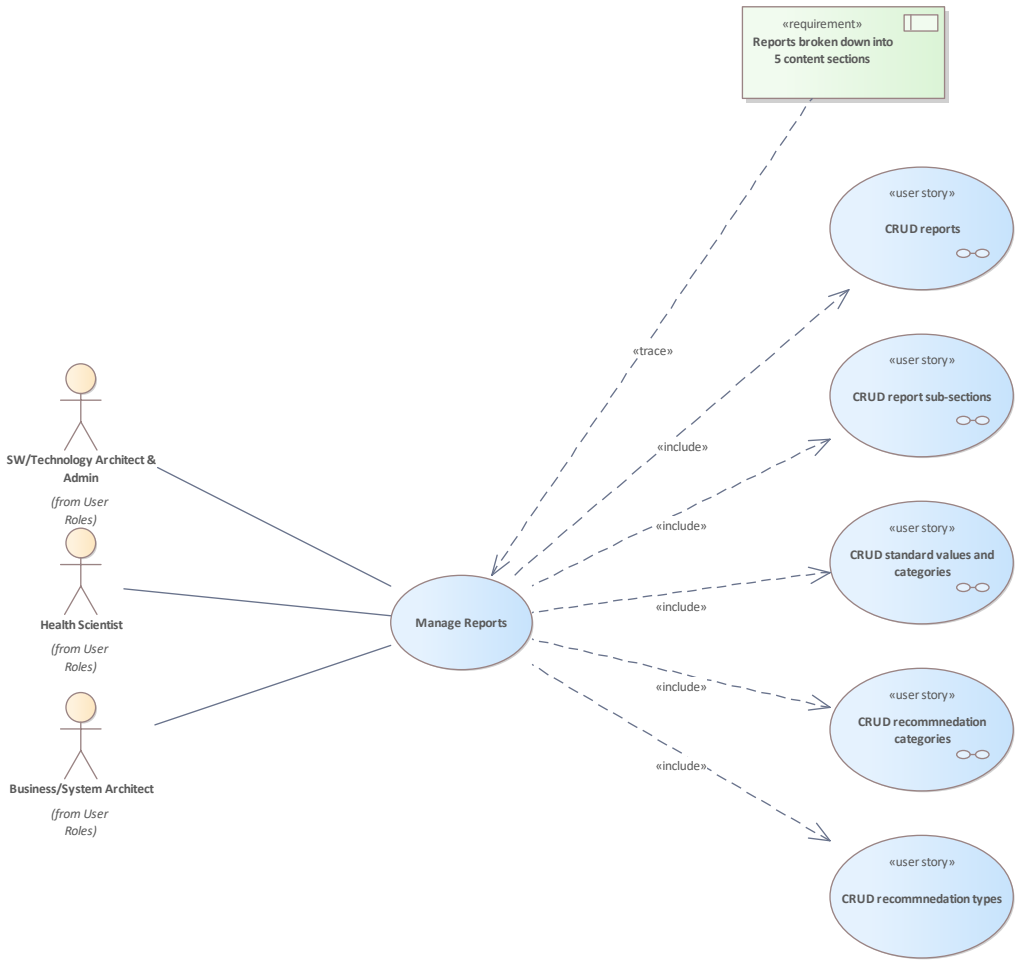


Figure 20: Manage Reports

1.1.3.7.1.1 CRUD recommendation categories diagram

Use Case diagram in package 'User stories (requirements) and use cases'

CRUD recommendation categories
Version 1.0
Author: Moe Huss

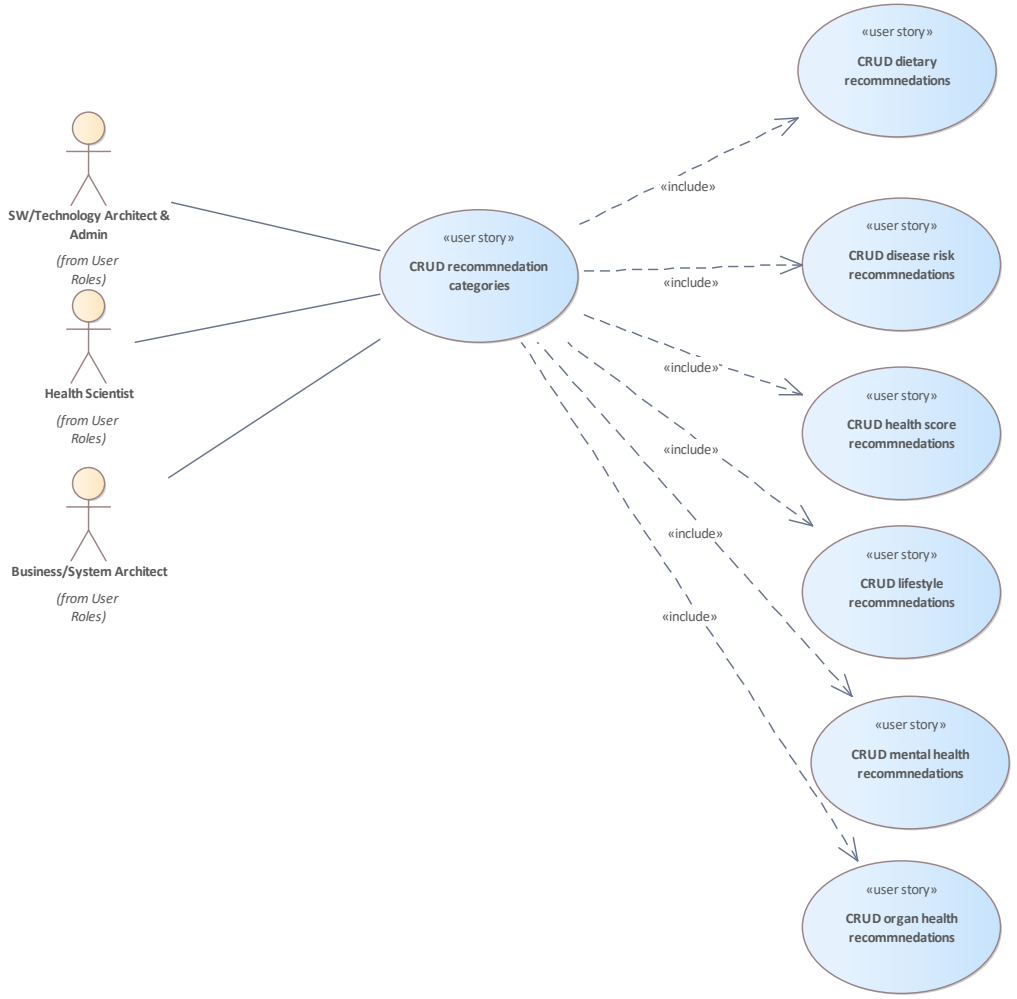


Figure 21: CRUD recommendation categories

1.1.3.7.1.2 CRUD report sub-sections diagram

Use Case diagram in package 'User stories (requirements) and use cases'

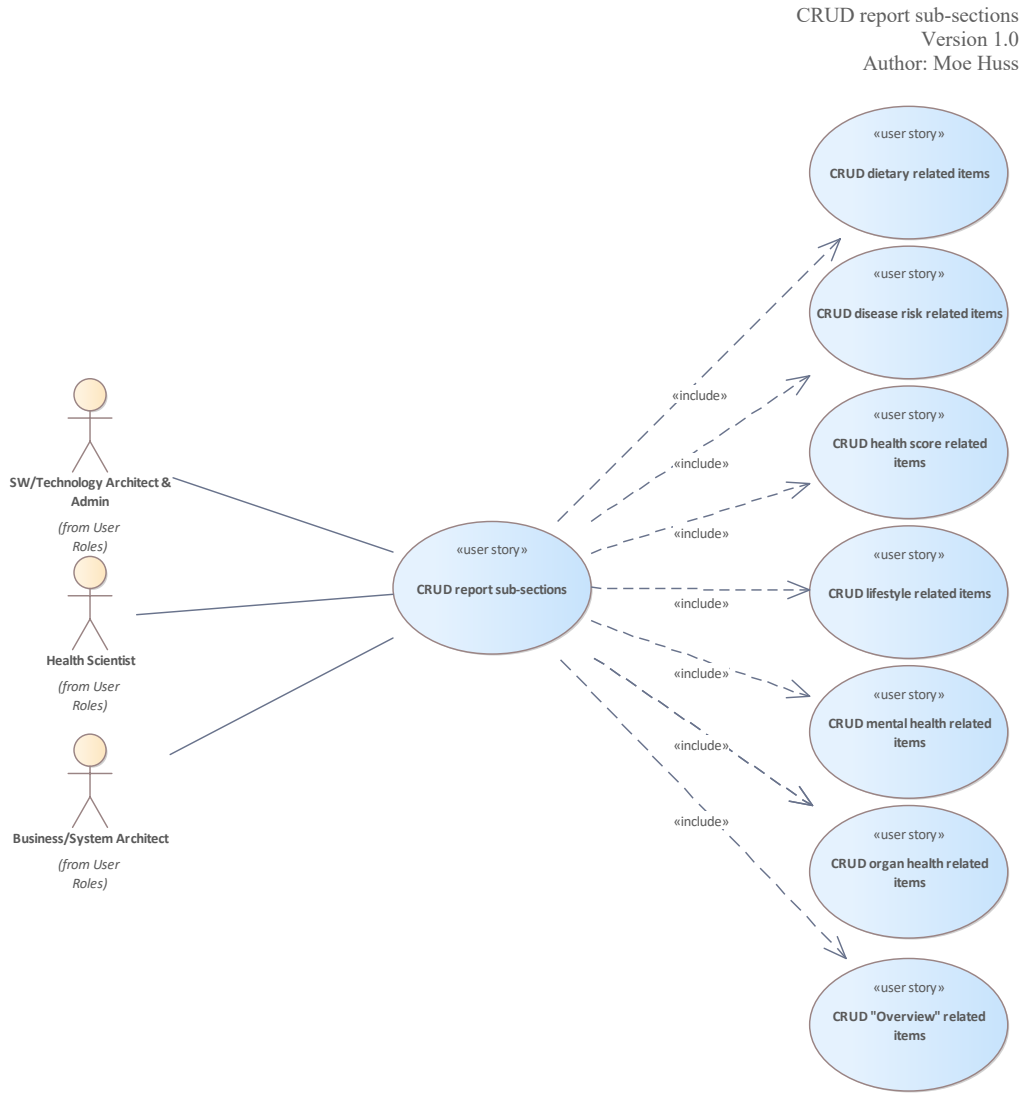


Figure 22: CRUD report sub-sections

1.1.3.7.1.3 CRUD reports diagram

Use Case diagram in package 'User stories (requirements) and use cases'

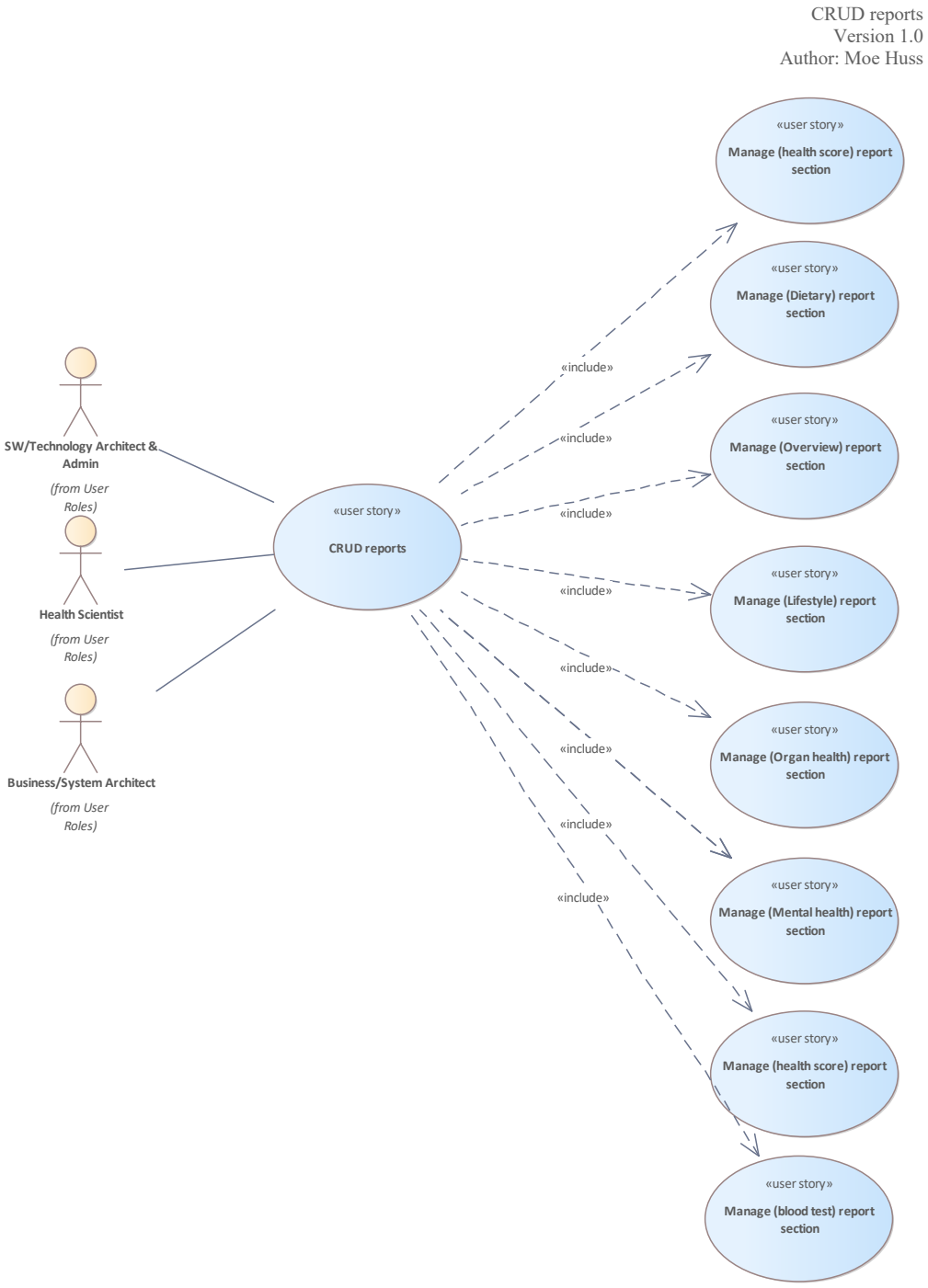


Figure 23: CRUD reports

1.1.3.7.1.4 CRUD standard values and categories diagram

Use Case diagram in package 'User stories (requirements) and use cases'

CRUD standard values and categories
Version 1.0
Author: Moe Huss

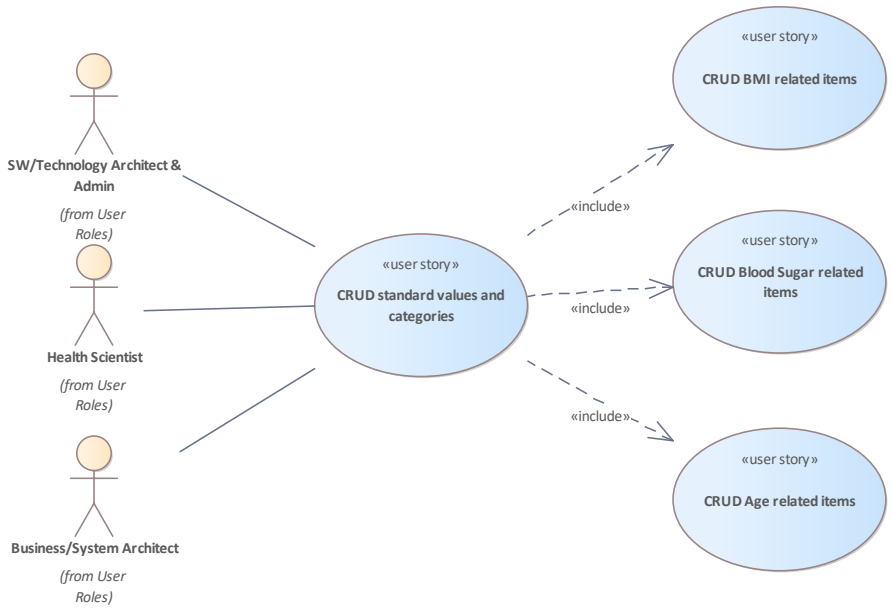


Figure 24: CRUD standard values and categories

1.1.3.7.2 Manage Surveys diagram

Use Case diagram in package 'User stories (requirements) and use cases'

Manage Surveys
Version 1.0
Author: Moe Huss

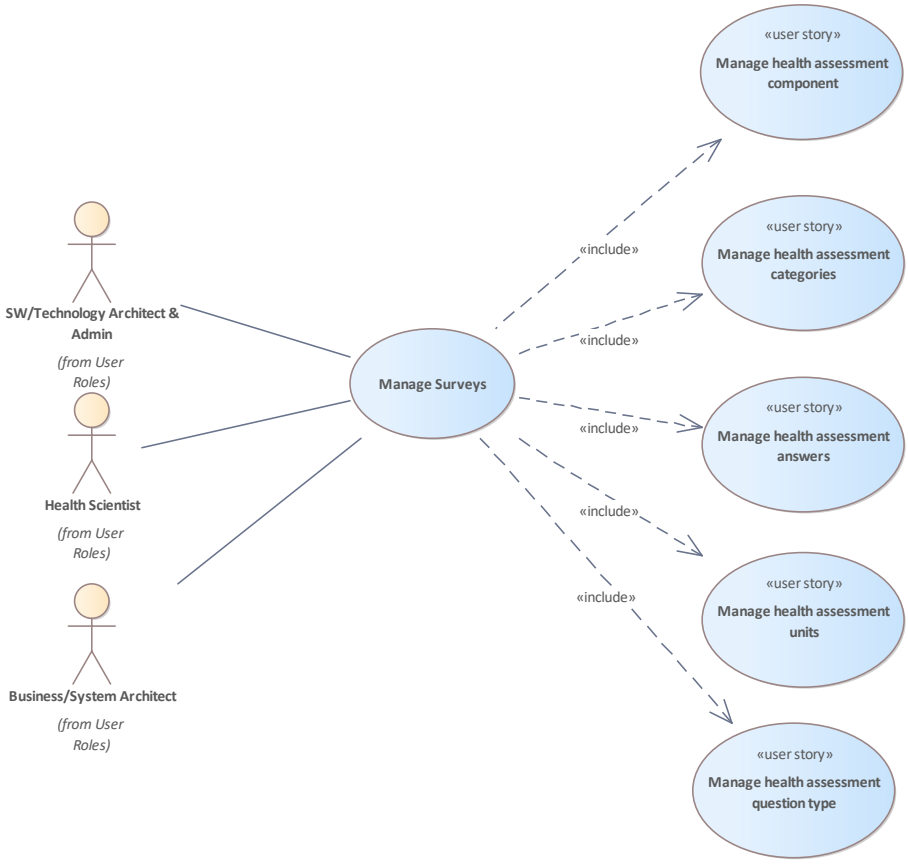


Figure 25: Manage Surveys

1.1.3.8 Epic Breakdown (Diet & Grocery Recommendation) diagram

Use Case diagram in package 'User stories (requirements) and use cases'

Epic Breakdown (Diet & Grocery Recommendation)
Version 1.0
Author: Moe Huss

Figure 26: Epic Breakdown (Diet & Grocery Recommendation)

1.1.3.8.1 Manage grocery list diagram

Use Case diagram in package 'User stories (requirements) and use cases'

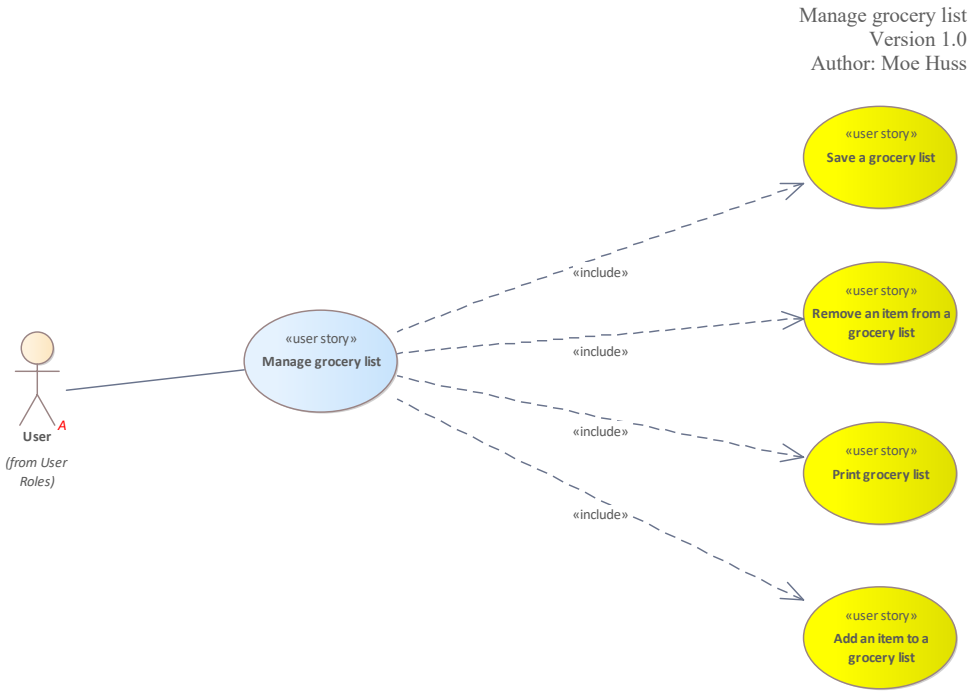


Figure 27: Manage grocery list

1.1.3.8.2 Educate user diagram

Use Case diagram in package 'User stories (requirements) and use cases'

Educate user
Version 1.0
Author: Moe Huss

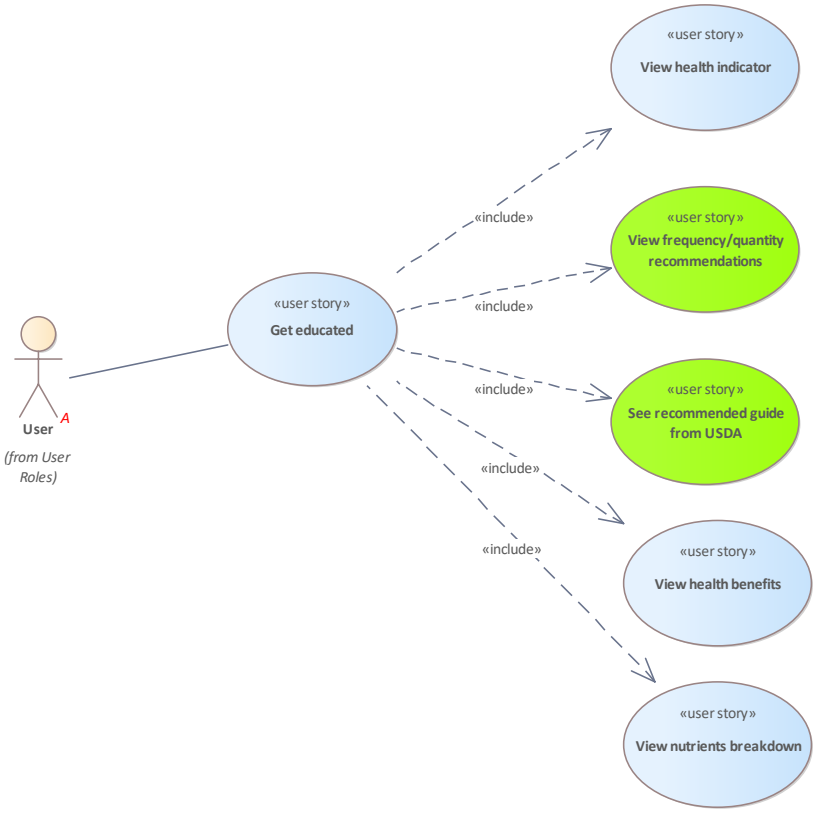


Figure 28: Educate user

1.1.3.8.3 Manage recepies diagram

Use Case diagram in package 'User stories (requirements) and use cases'

Manage recepies
Version 1.0
Author: Moe Huss

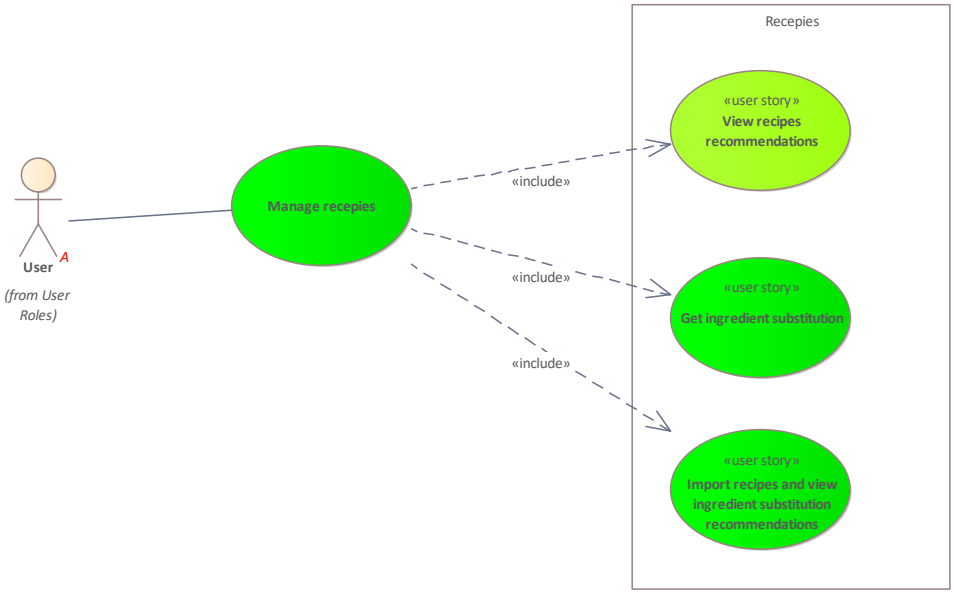


Figure 29: Manage recepies

1.1.3.9 Diet Tracker diagram

Use Case diagram in package 'User stories (requirements) and use cases'

Diet Tracker
Version 1.0
Author: Moe Huss

Figure 30: Diet Tracker

1.1.3.10 Health Monitor (or Nutrition Analyzer) diagram

Use Case diagram in package 'User stories (requirements) and use cases'

Health Monitor (or Nutrition Analyzer)
Version 1.0
Author: Moe Huss

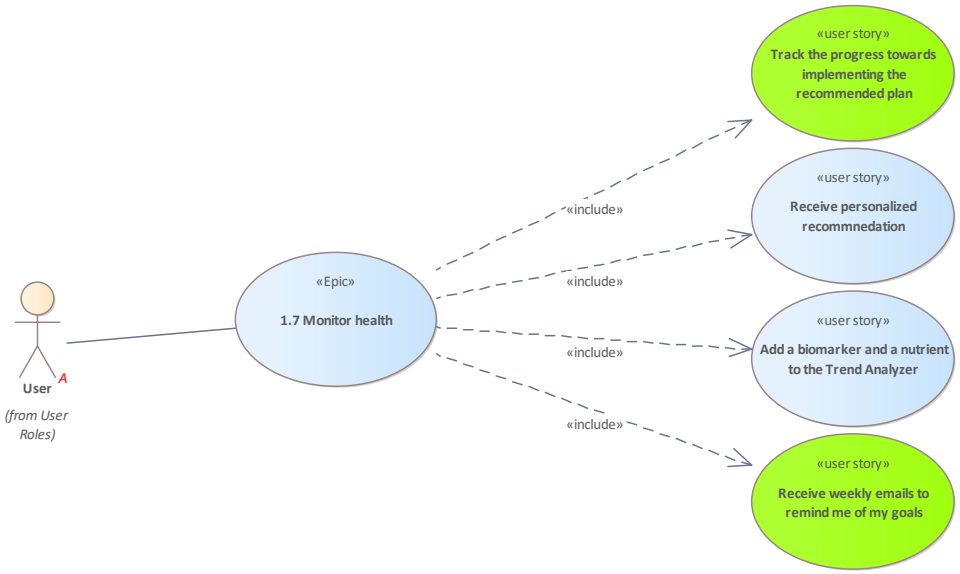


Figure 31: Health Monitor (or Nutrition Analyzer)

Diagram Report

b. Structure

Version 1.0 • Final

Date Generated: 2/21/2022

Author: Moe Huss

CREATED WITH ENTERPRISE ARCHITECT

Table of Contents

1.1.1 Conceptual Domain Diagram diagram

Class diagram in package 'Conceptual Structure'

Conceptual Domain Diagram
Version 1.0
Author: Moe Huss

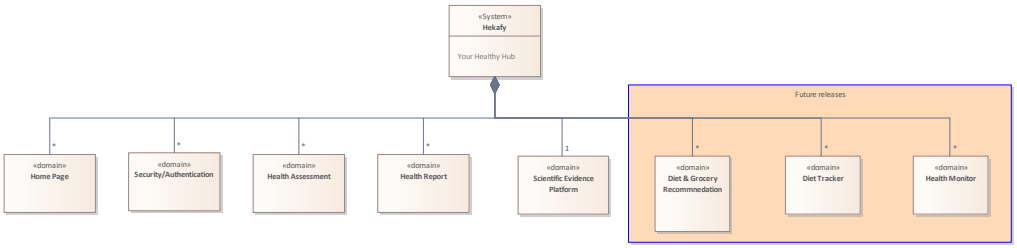


Figure 1: Conceptual Domain Diagram

1.1.2 Application Layers diagram

Logical diagram in package 'Application Layer'

Application Layers
Version 1.0
Author: Moe Huss

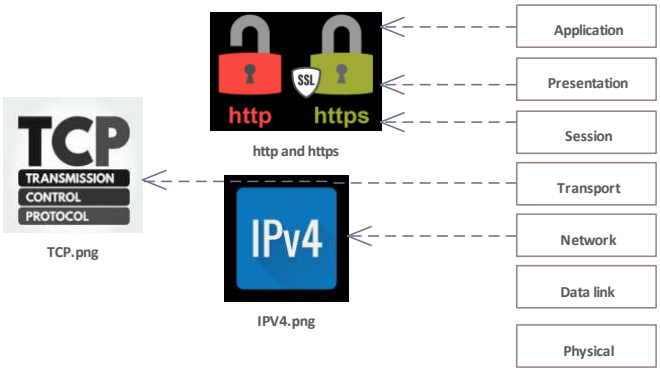


Figure 2: Application Layers

1.1.3 Logical Application Layers diagram

Logical diagram in package 'Application Layer'

Logical Application Layers
Version 1.0
Author: Moe Huss

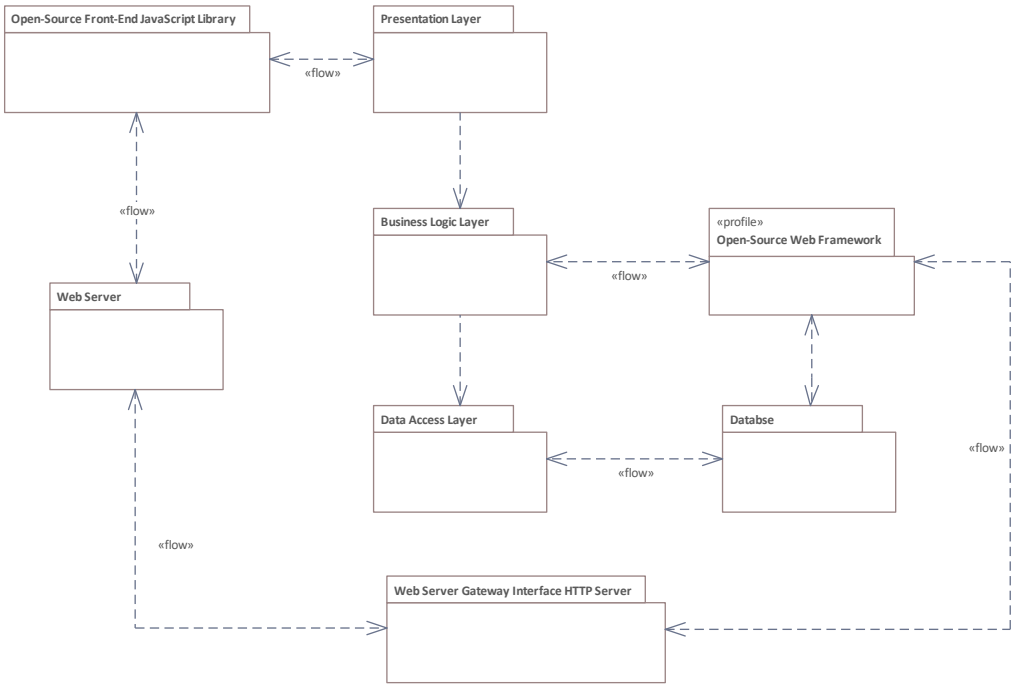


Figure 3: Logical Application Layers

1.1.4 Physical Application Layers diagram

Logical diagram in package 'Application Layer'

Physical Application Layers
Version 1.0
Author: Moe Huss

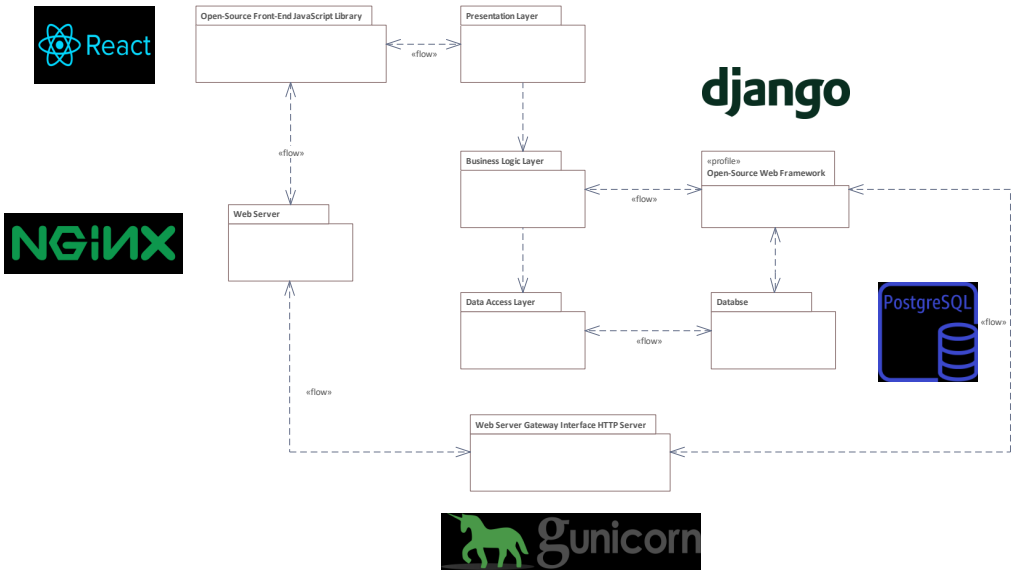


Figure 4: Physical Application Layers

1.1.5 Physical Structure diagram

Component diagram in package 'Physical Structure - Instances on AWS'

Physical Structure
Version 1.0
Author: Moe Huss

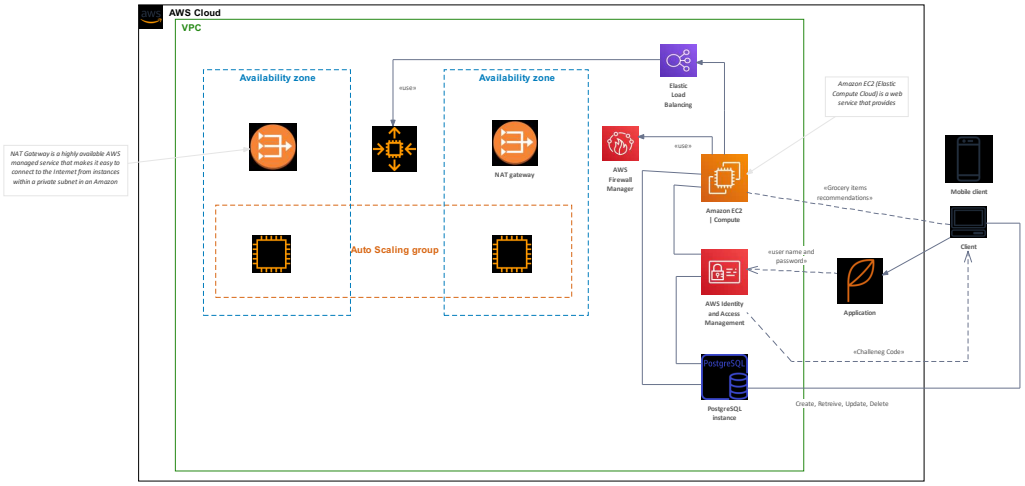


Figure 5: Physical Structure

1.1.6 Physical Structure - Instances on AWS diagram

Class diagram in package 'Physical Structure - Instances on AWS'

Physical Structure - Instances on AWS
Version 1.0
Author: Moe Huss

Backend implementation of the main system domains

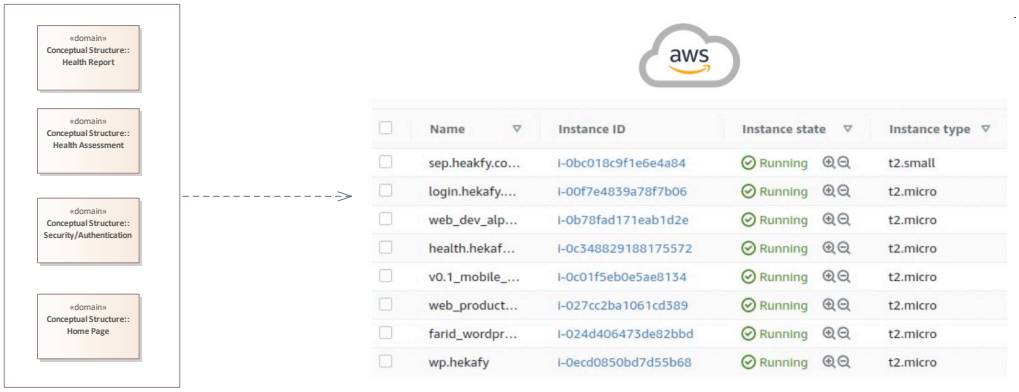


Figure 6: Physical Structure - Instances on AWS

1.1.7 Physical Structure of AWS implementation diagram

Component diagram in package 'Physical Structure - Instances on AWS'

Physical Structure of AWS implementation
Version 1.0
Author: Moe Huss

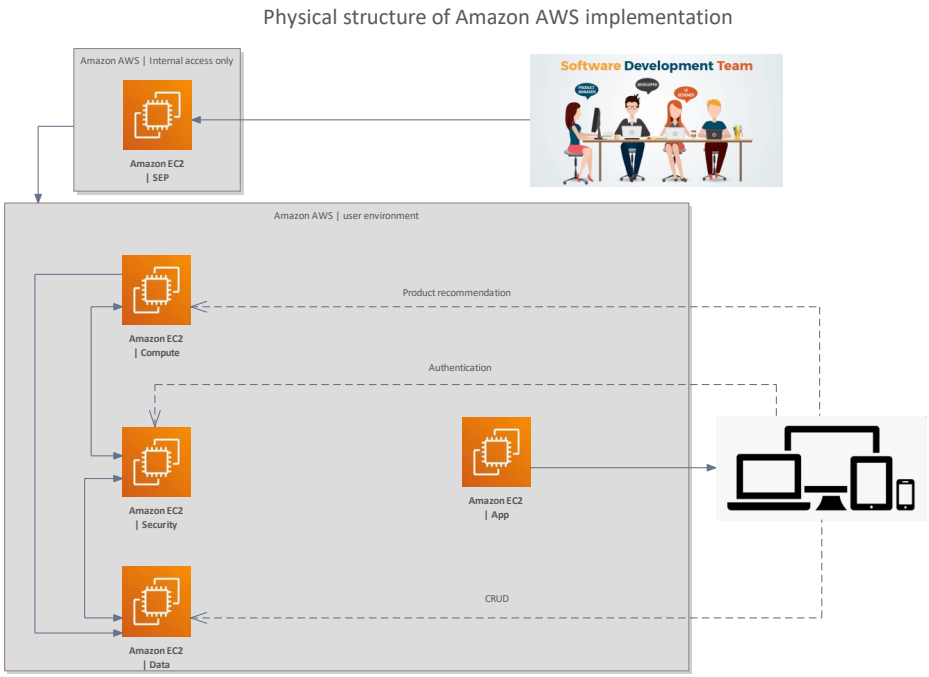


Figure 7: Physical Structure of AWS implementation

1.1.8 Physical Structure of the environment diagram

Component diagram in package 'Physical Structure - Instances on AWS'

Physical Structure of the environment
Version 1.0
Author: Moe Huss

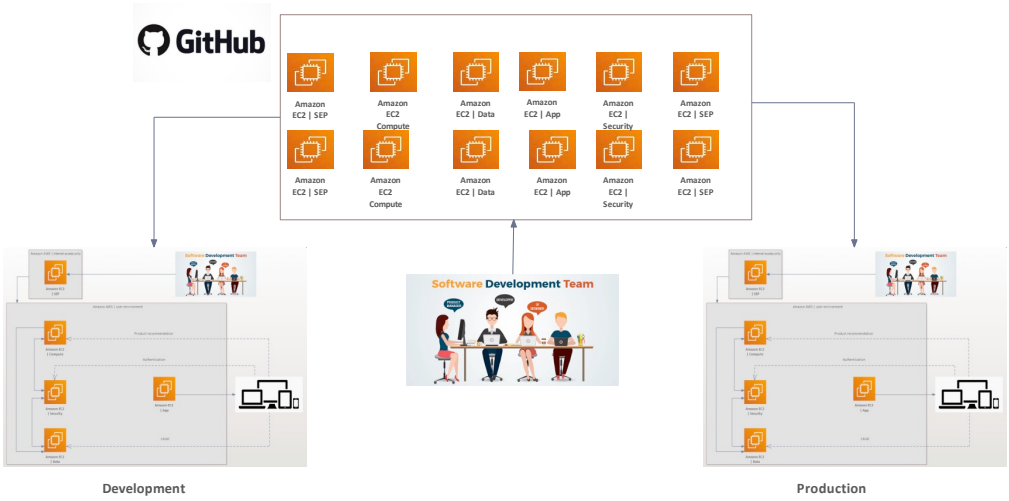


Figure 8: Physical Structure of the environment

Diagram Report

c. Behavior

Version 1.0 • Final

Date Generated: 2/21/2022

Author: Moe Huss

CREATED WITH ENTERPRISE ARCHITECT

Table of Contents

1.1.1 Dietary habits questions flow diagram

Activity diagram in package 'Dietary habits questions flow'

Dietary habits questions flow
Version 1.0
Author: Moe Huss

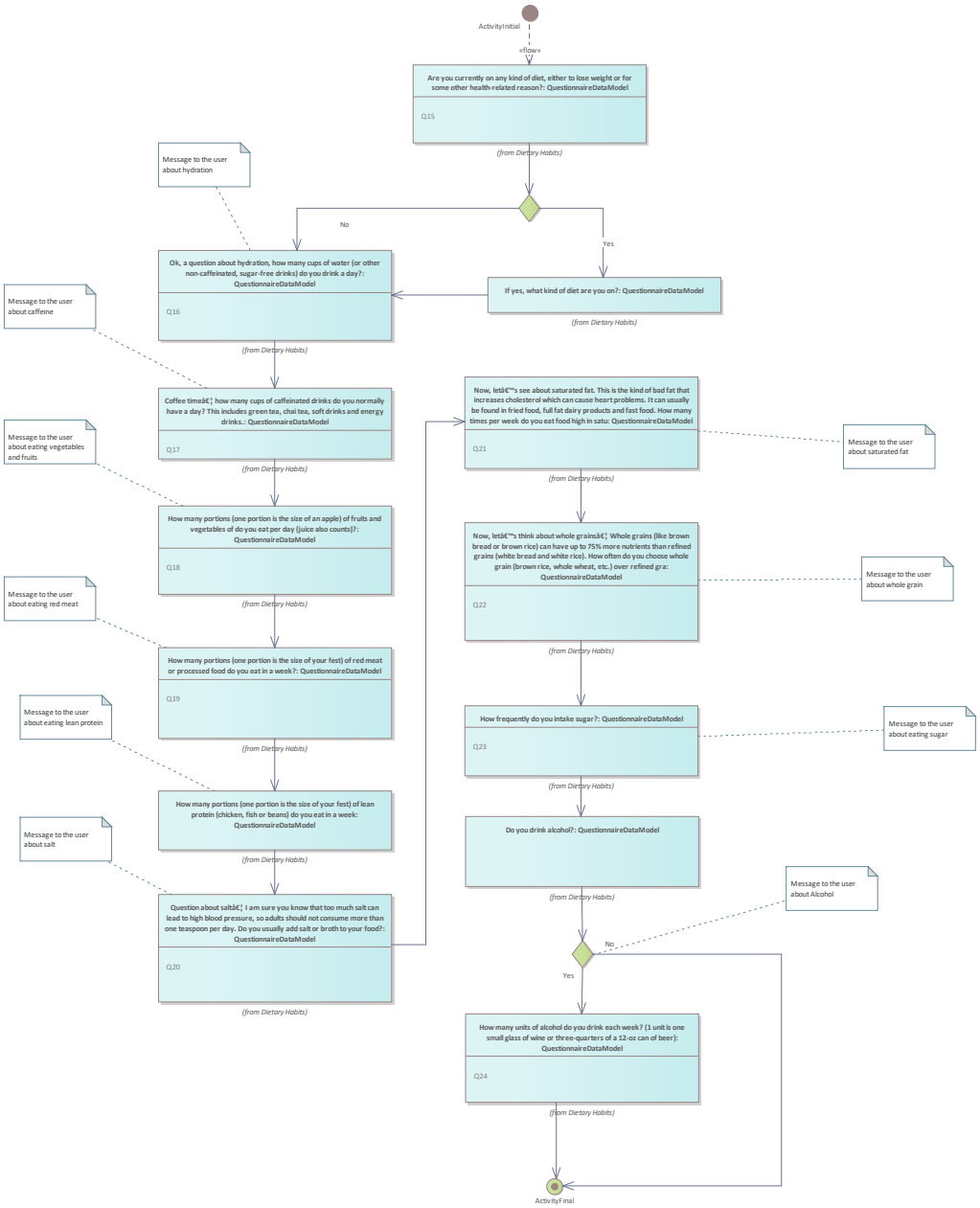


Figure 1: Dietary habits questions flow

1.1.2 Lifestyle info questions flow diagram

Activity diagram in package 'Lifestyle info questions flow'

Lifestyle info questions flow
Version 1.0
Author: Moe Huss

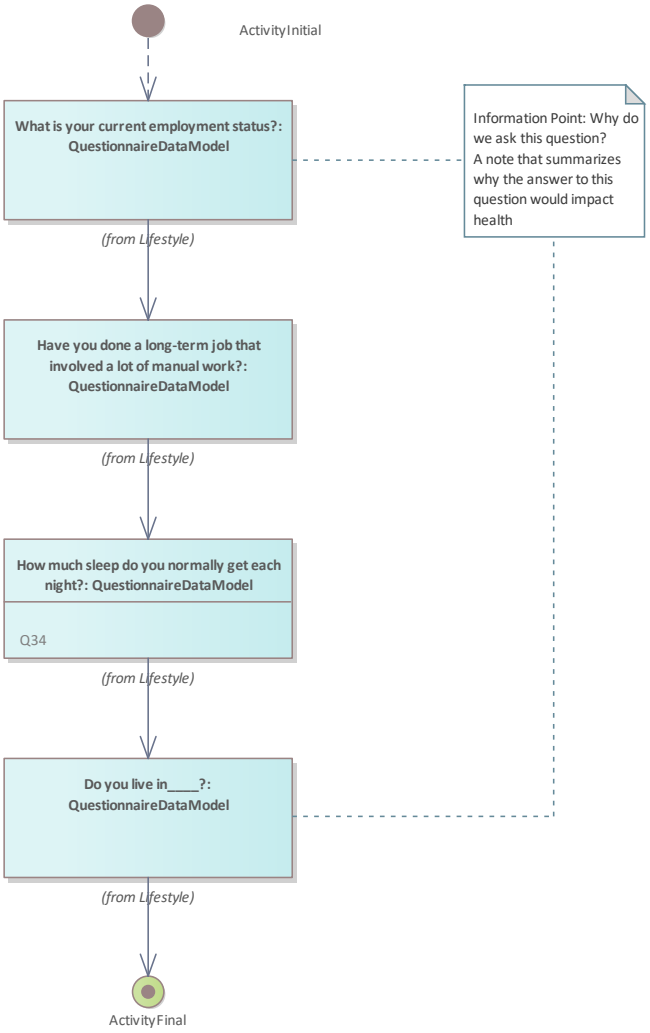


Figure 2: Lifestyle info questions flow

1.1.3 Basic info questions flow diagram

Activity diagram in package 'Basic info questions flow'

Basic info questions flow
Version 1.0
Author: Moe Huss

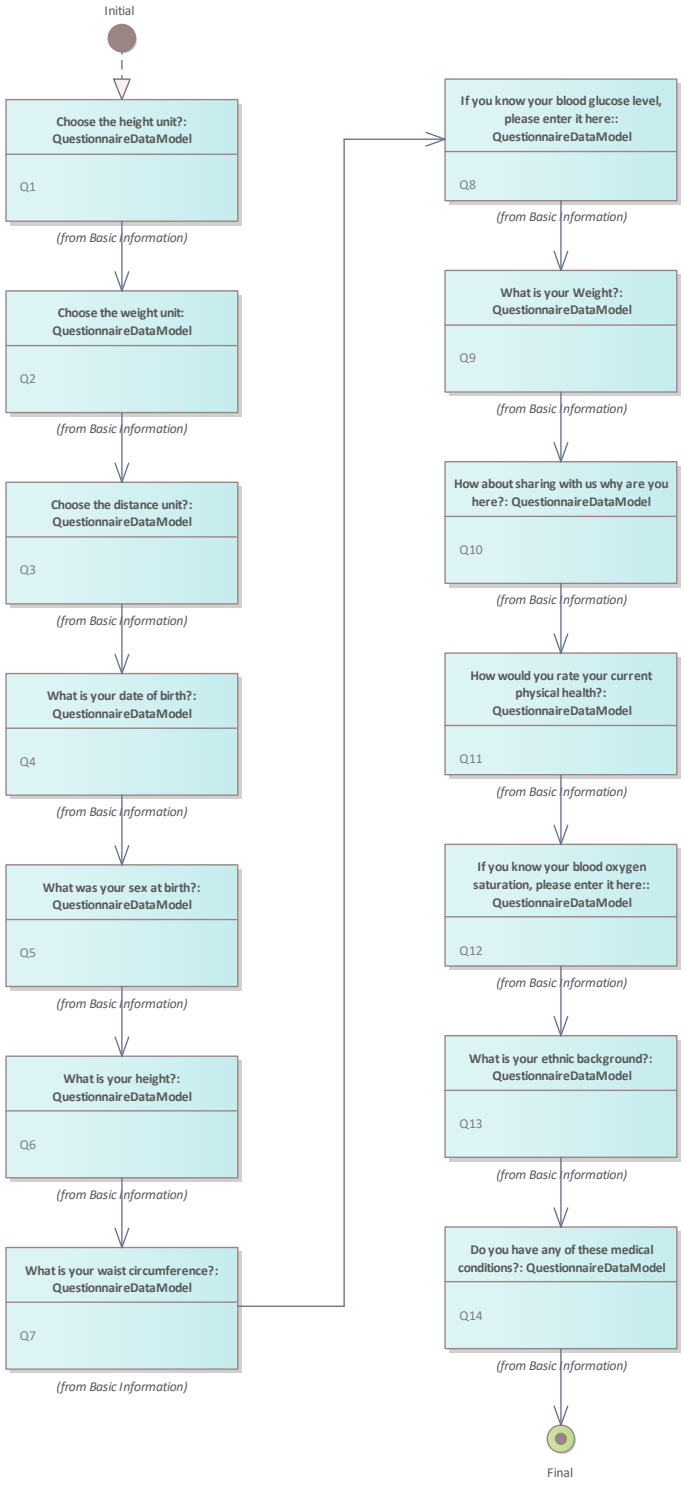


Figure 3: Basic info questions flow

Diagram Report

d. Data

Version 1.0 • Final

Date Generated: 2/21/2022

Author: Moe Huss

CREATED WITH ENTERPRISE ARCHITECT

Table of Contents

1.1.1 Conceptual Model v1.0 diagram

Class diagram in package 'Conceptual Data Model_v1.0'

Conceptual Model v1.0
Version 1.0
Author: Moe Huss

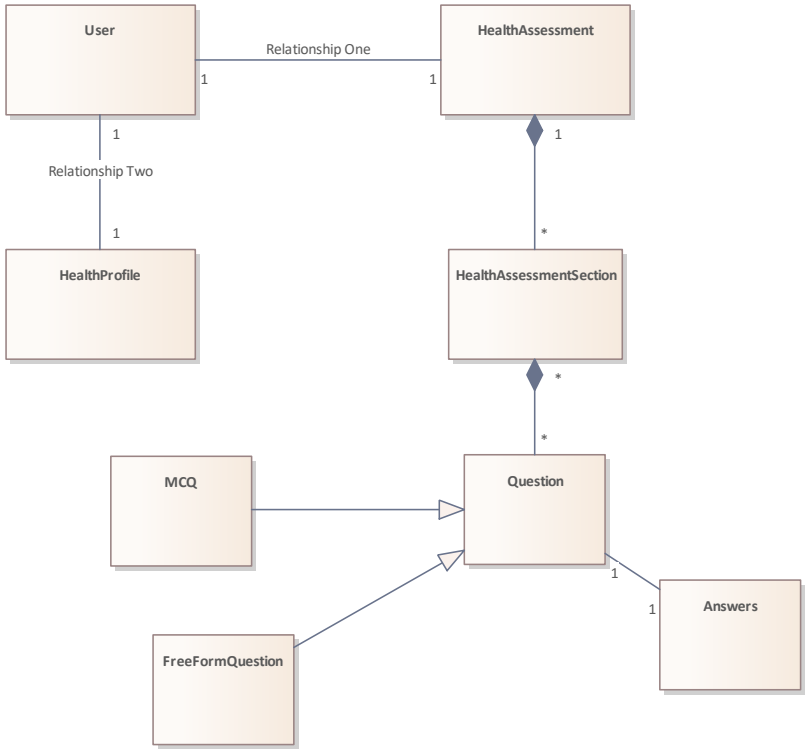


Figure 1: Conceptual Model v1.0

1.1.2 Logical Model v1.0 diagram

Class diagram in package 'Logical Data Model_v1.0'

Logical Model v1.0
Version 1.0
Author: Moe Huss

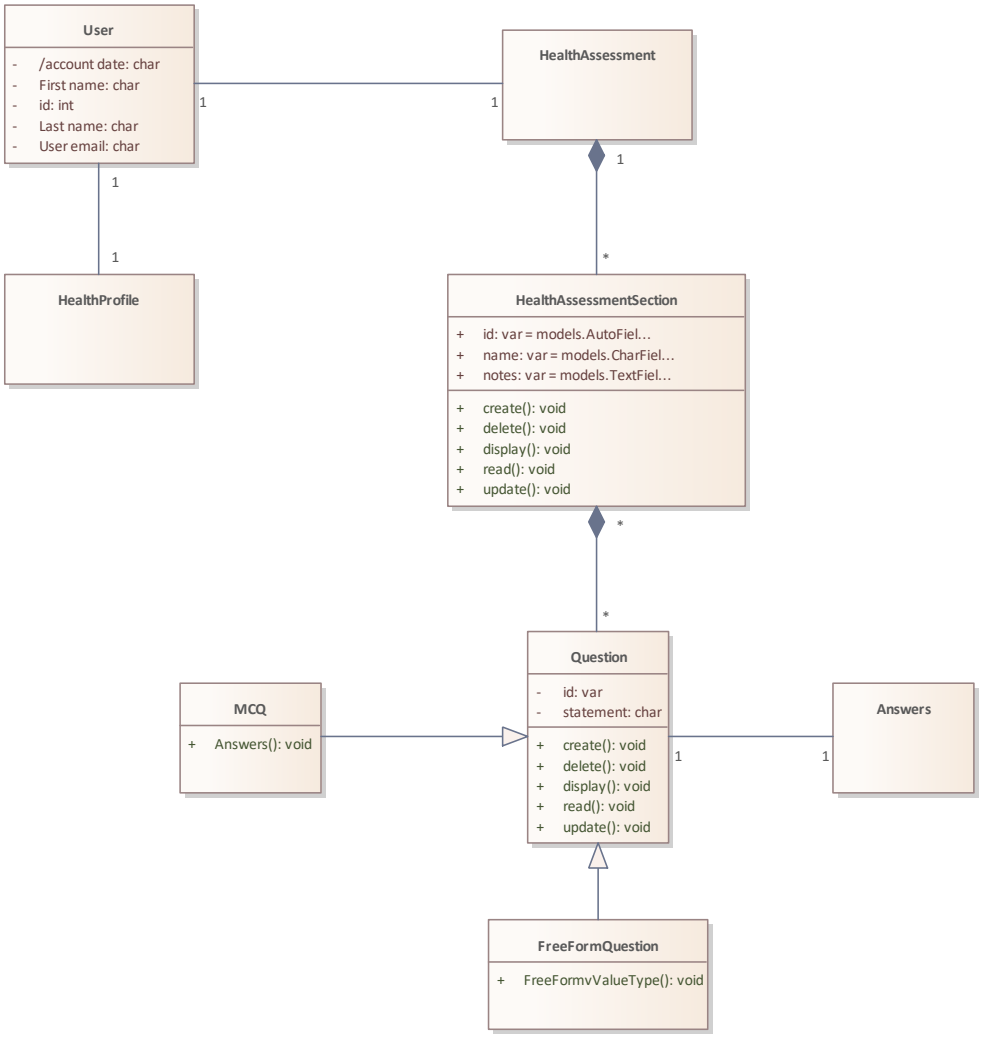


Figure 2: Logical Model v1.0

1.1.3 Conceptual Model v2 diagram

Class diagram in package 'Conceptual Data Model v2'

Conceptual Model v2
Version
Author: Moe Huss

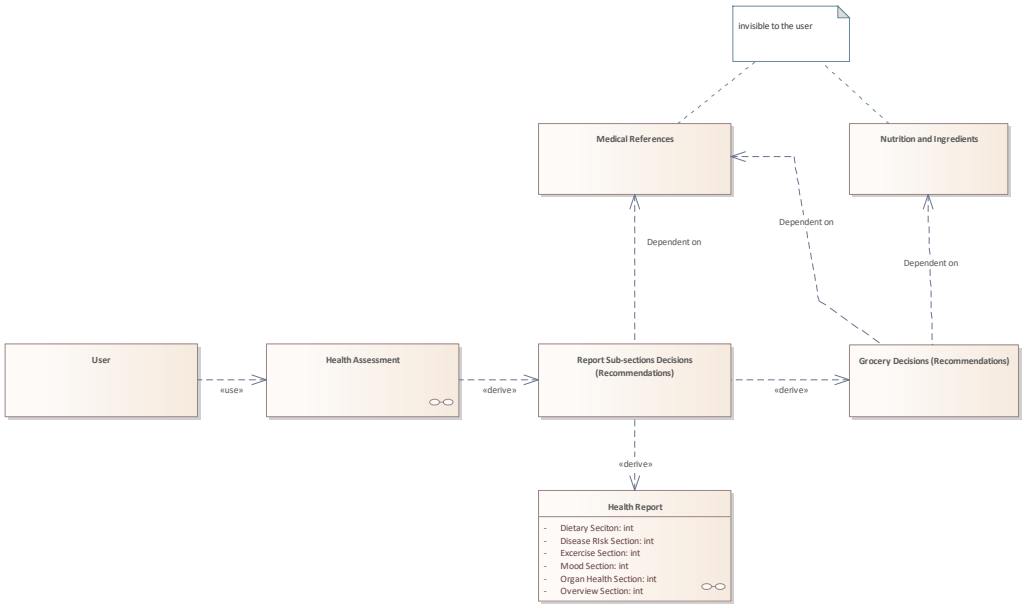


Figure 3: Conceptual Model v2

1.1.3.1 Health Assessment CDM diagram

Class diagram in package 'Conceptual Data Model v2'

Health Assessment CDM
Version 1.0
Author: Moe Huss

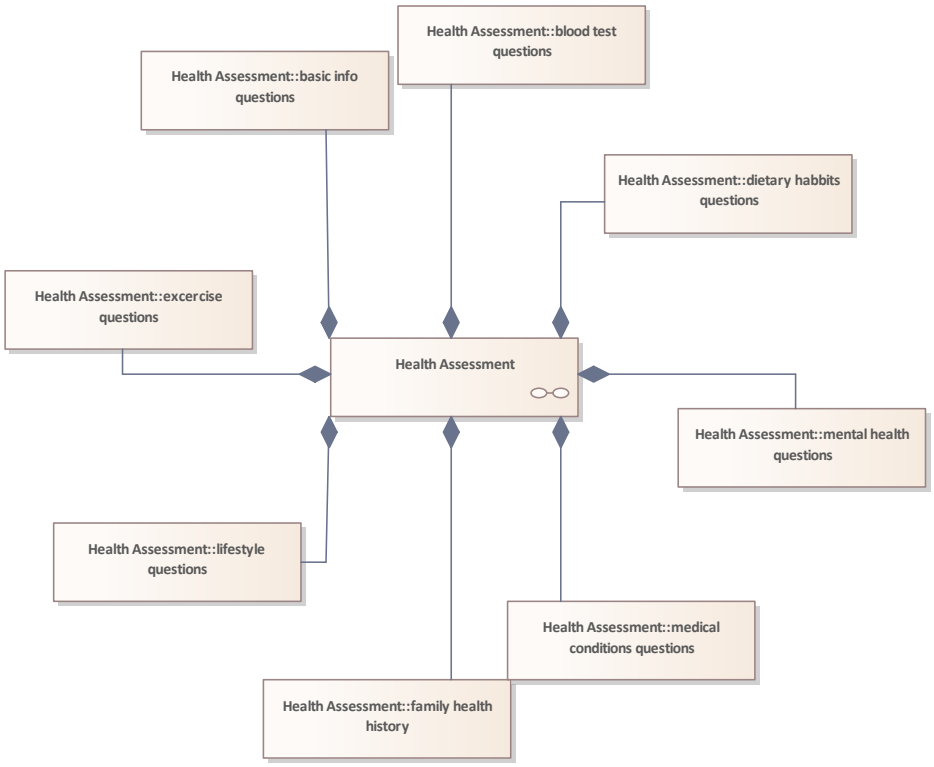


Figure 4: Health Assessment CDM

1.1.3.2 Health Report CDM diagram

Class diagram in package 'Conceptual Data Model v2'

Health Report CDM
Version 1.0
Author: Moe Huss

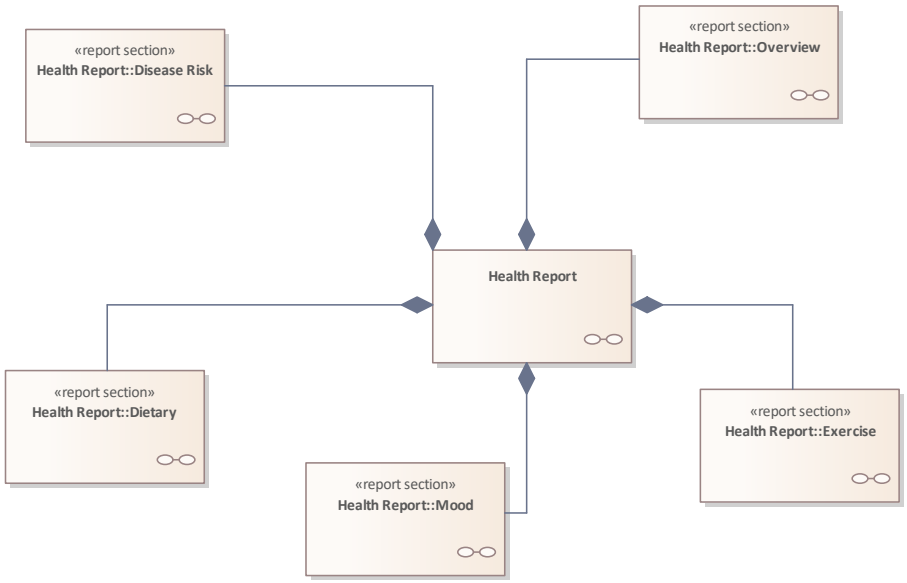


Figure 5: Health Report CDM

1.1.3.2.1 Overview diagram

Class diagram in package 'Conceptual Data Model v2'

Overview
Version 1.0
Author: Moe Huss

Figure 6: Overview

1.1.3.2.2 Disease Risk diagram

Class diagram in package 'Conceptual Data Model v2'

Disease Risk
Version 1.0
Author: Moe Huss

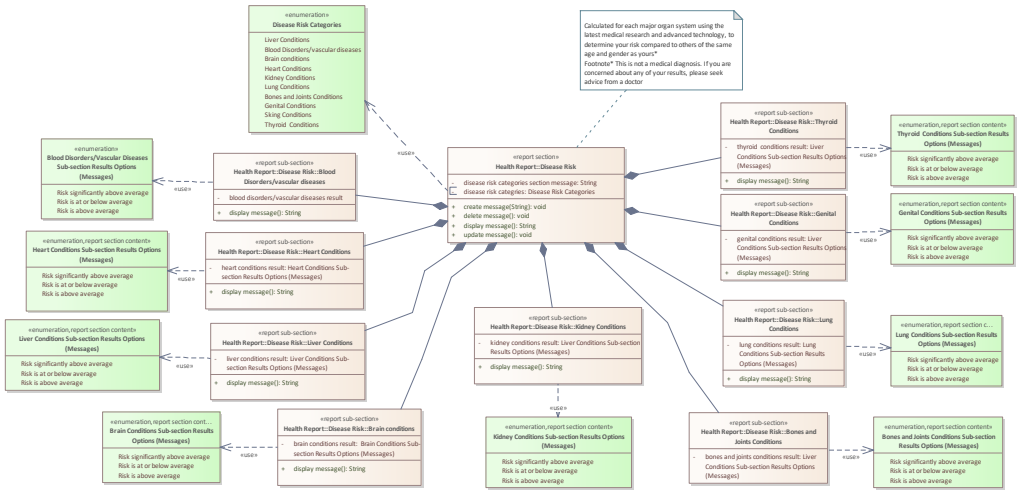


Figure 7: Disease Risk

1.1.3.2.3 Organ Health diagram

Class diagram in package 'Conceptual Data Model v2'

Organ Health
Version 1.0
Author: Moe Huss

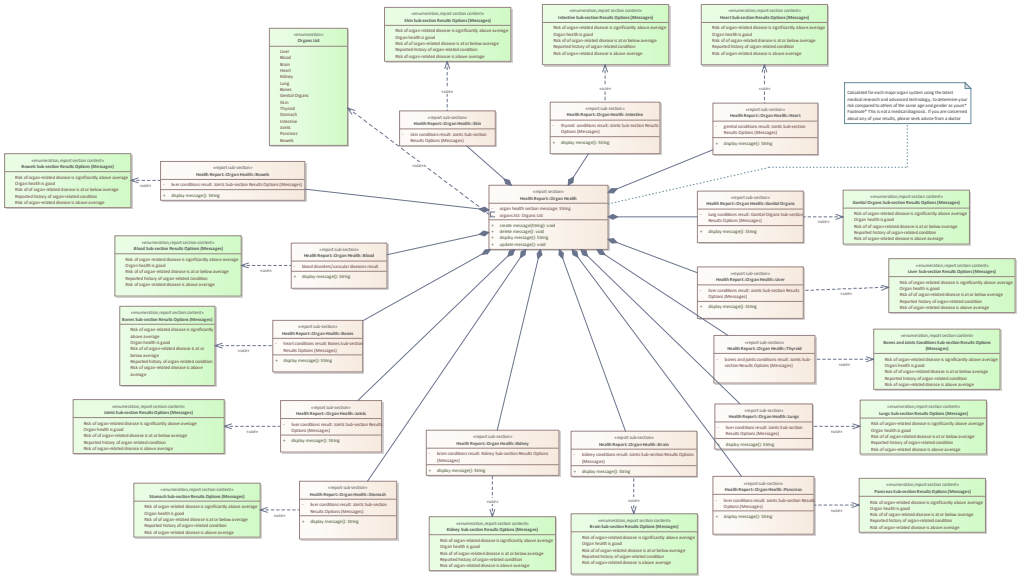


Figure 8: Organ Health

1.1.3.2.4 Dietary diagram

Class diagram in package 'Conceptual Data Model v2'

Dietary
Version 1.0
Author: Moe Huss

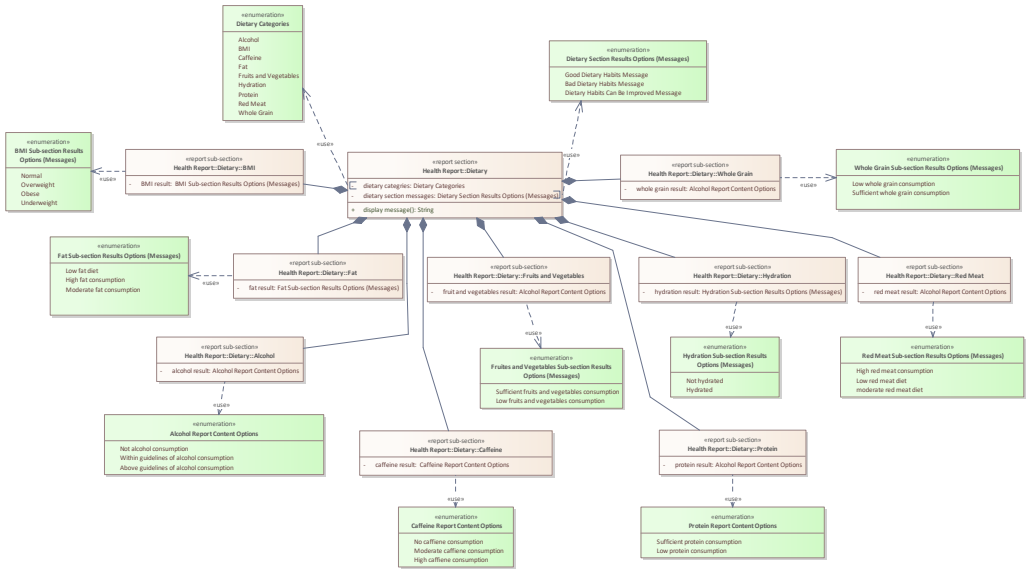


Figure 9: Dietary

1.1.3.2.5 Excercise diagram

Class diagram in package 'Conceptual Data Model v2'

Excercise
Version 1.0
Author: Moe Huss

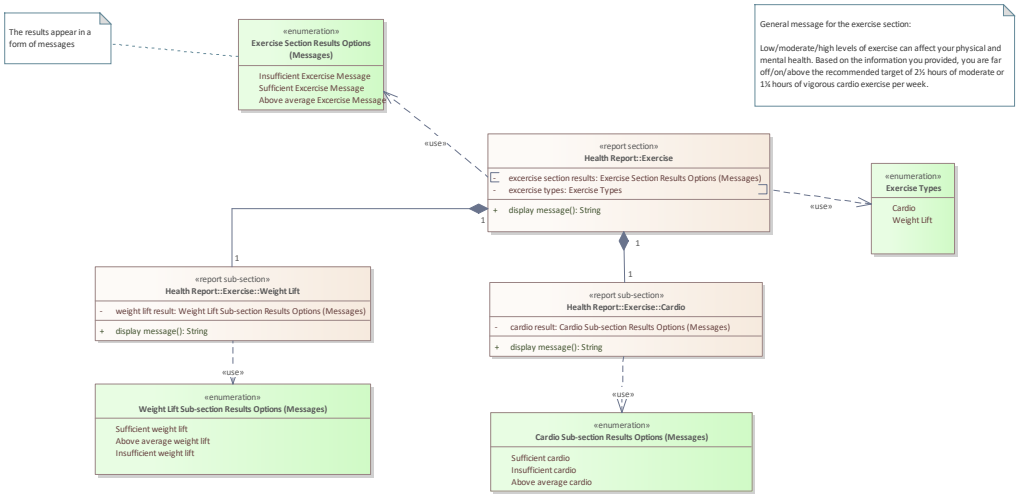


Figure 10: Excercise

1.1.3.2.6 Mood diagram

Class diagram in package 'Conceptual Data Model v2'

Mood
Version 1.0
Author: Moe Huss

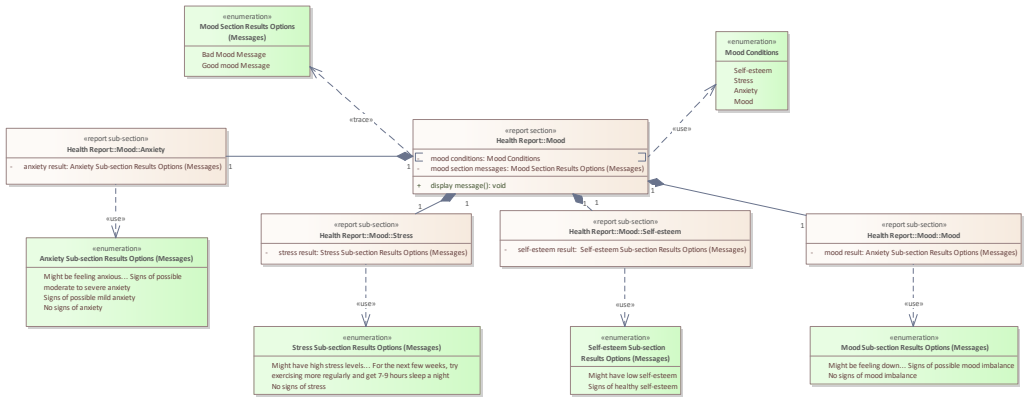


Figure 11: Mood

1.1.4 Logical Model v2 diagram

Class diagram in package 'Logical Data Model v2'

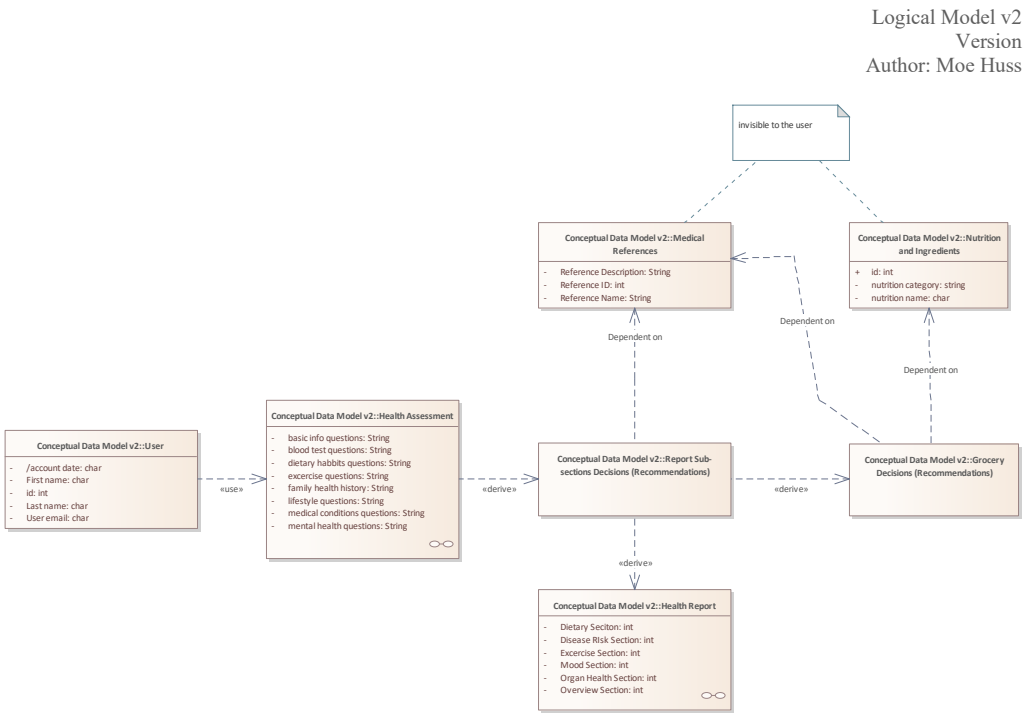


Figure 12: Logical Model v2

1.1.4.1.1 Health Assessment diagram

Class diagram in package 'Health Assessment'

Health Assessment
Version 1.0
Author: Moe Huss

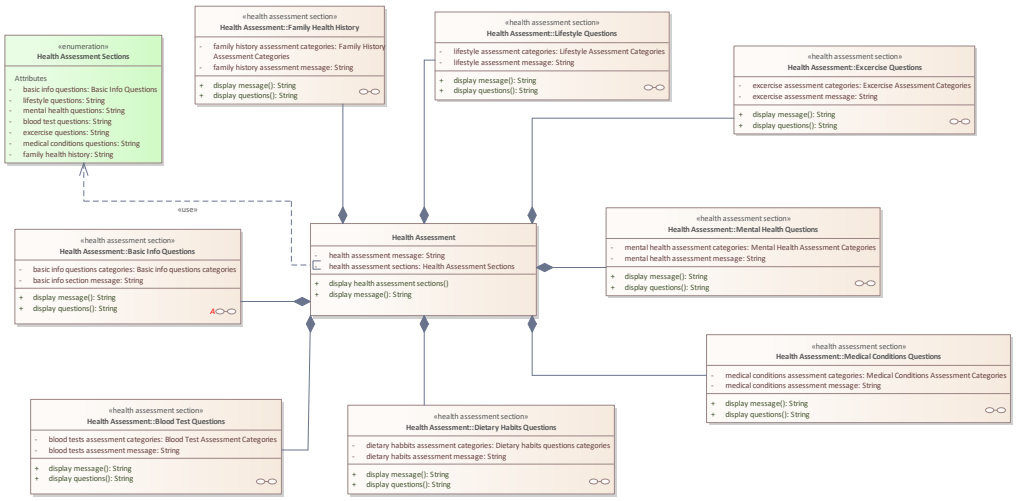


Figure 13: Health Assessment

1.1.4.1.1.1 basic info questions diagram

Class diagram in package 'Health Assessment'

basic info questions
Version 1.0
Author: Moe Huss

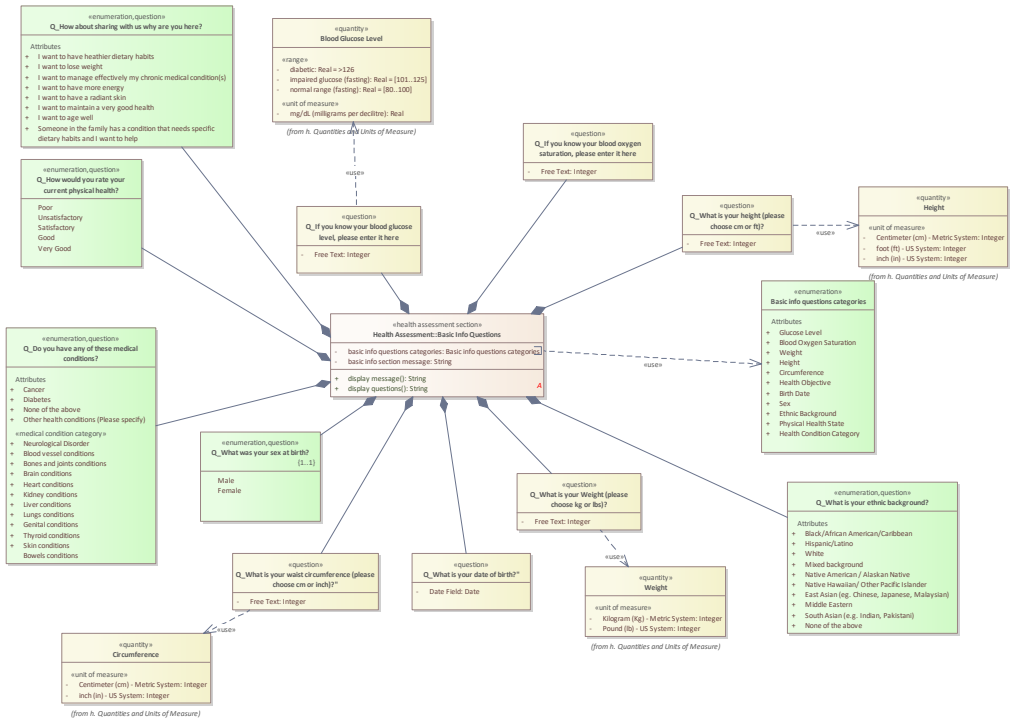


Figure 14: basic info questions

1.1.4.1.1.2 dietary habits questions diagram

Class diagram in package 'Health Assessment'

dietary habits questions
Version 1.0
Author: Moe Huss

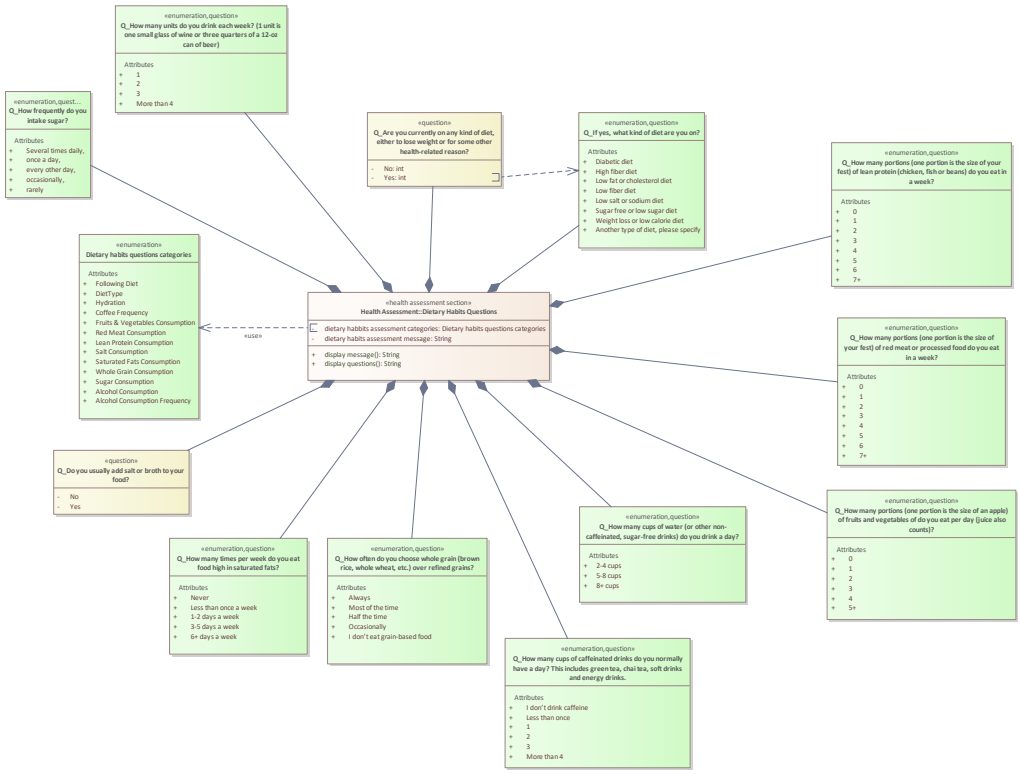


Figure 15: dietary habits questions

1.1.4.1.1.3 exercise questions diagram

Class diagram in package 'Health Assessment'

exercise questions
Version 1.0
Author: Moe Huss

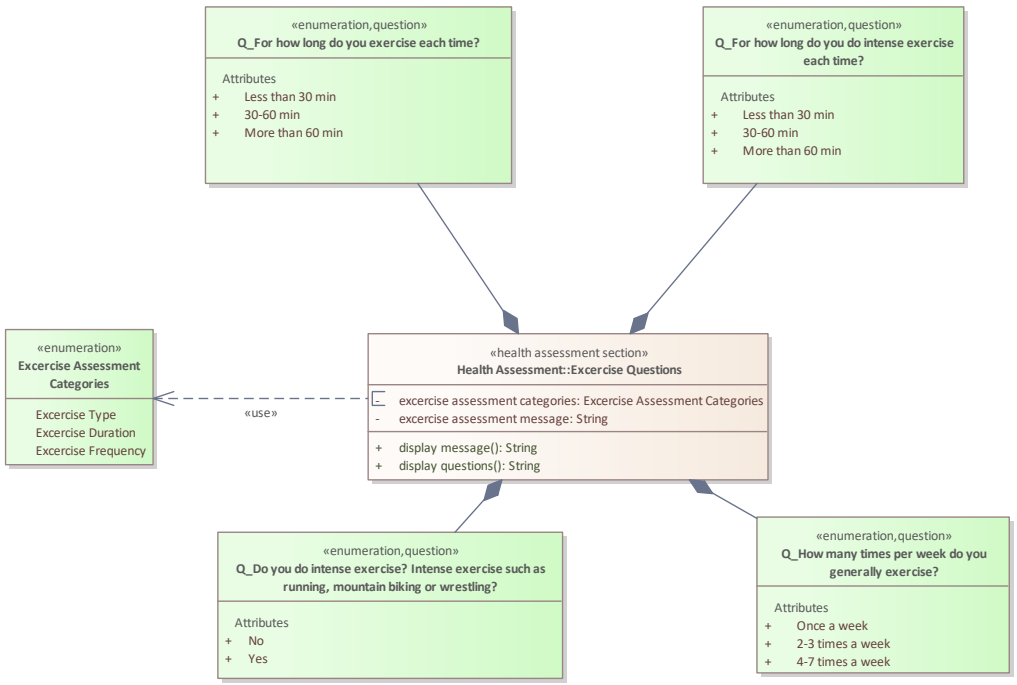


Figure 16: exercise questions

1.1.4.1.1.4 medical conditions questions diagram

Class diagram in package 'Health Assessment'

medical conditions questions
Version 1.0
Author: Moe Huss

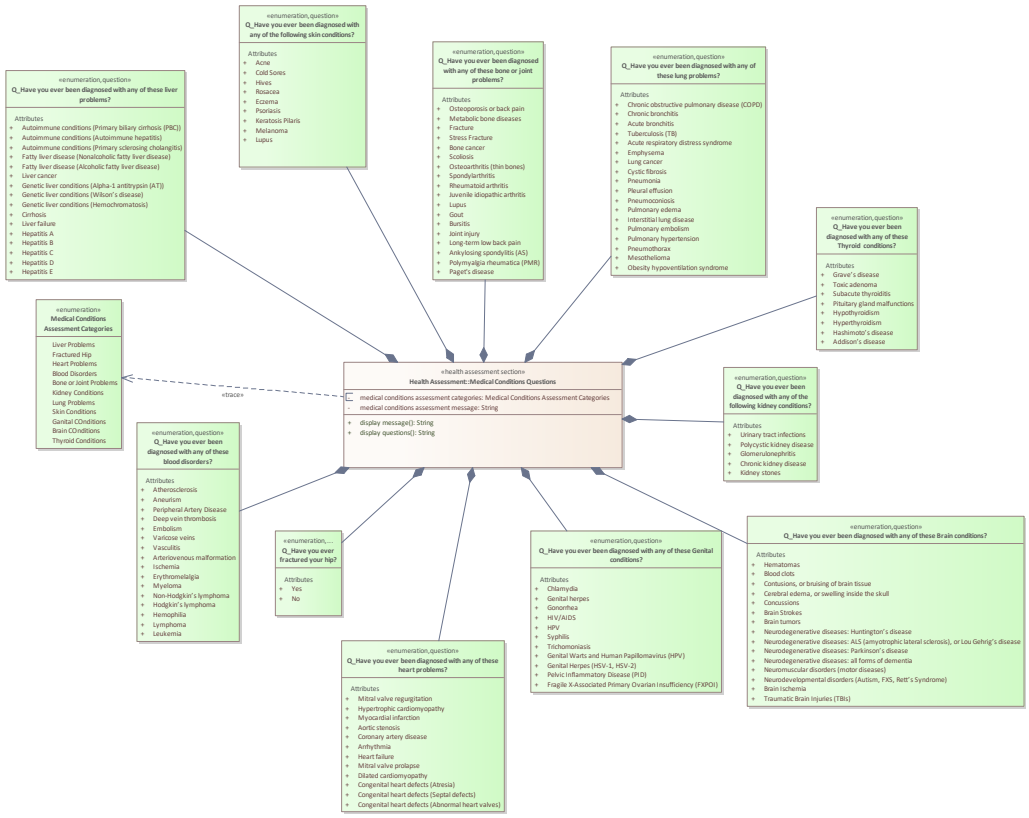


Figure 17: medical conditions questions

1.1.4.1.1.5 family health history diagram

Class diagram in package 'Health Assessment'

family health history
Version 1.0
Author: Moe Huss

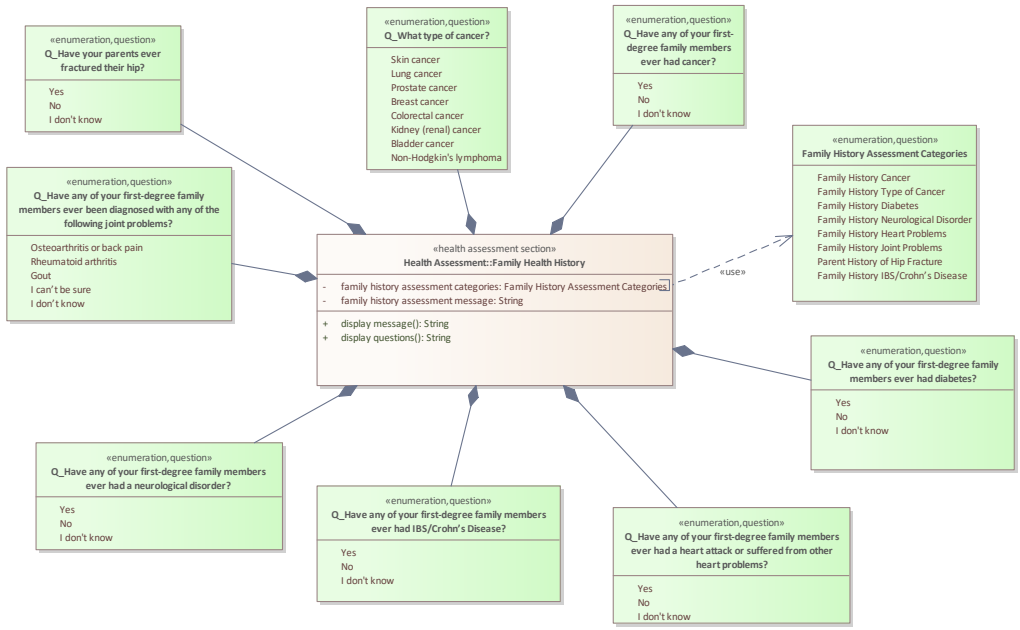


Figure 18: family health history

1.1.4.1.1.6 lifestyle questions diagram

Class diagram in package 'Health Assessment'

lifestyle questions
Version 1.0
Author: Moe Huss

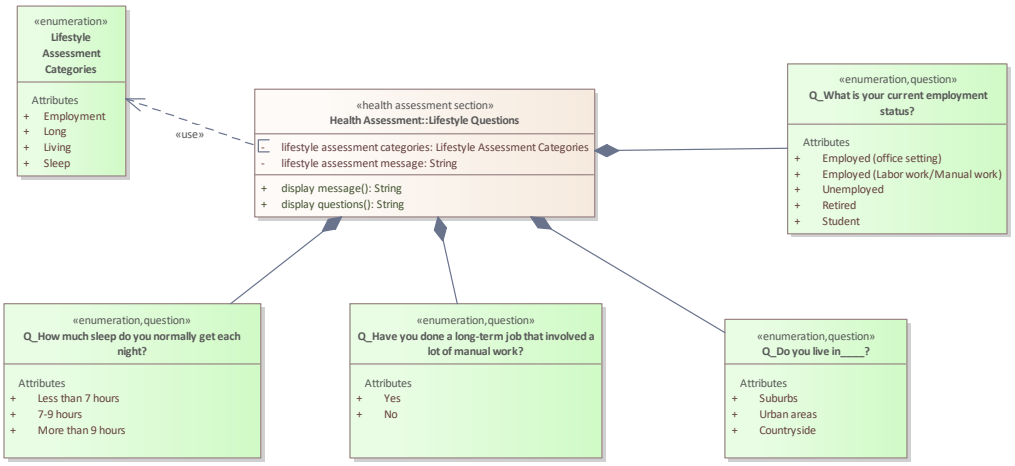


Figure 19: lifestyle questions

1.1.4.1.1.7 mental health questions diagram

Class diagram in package 'Health Assessment'

mental health questions
Version 1.0
Author: Moe Huss

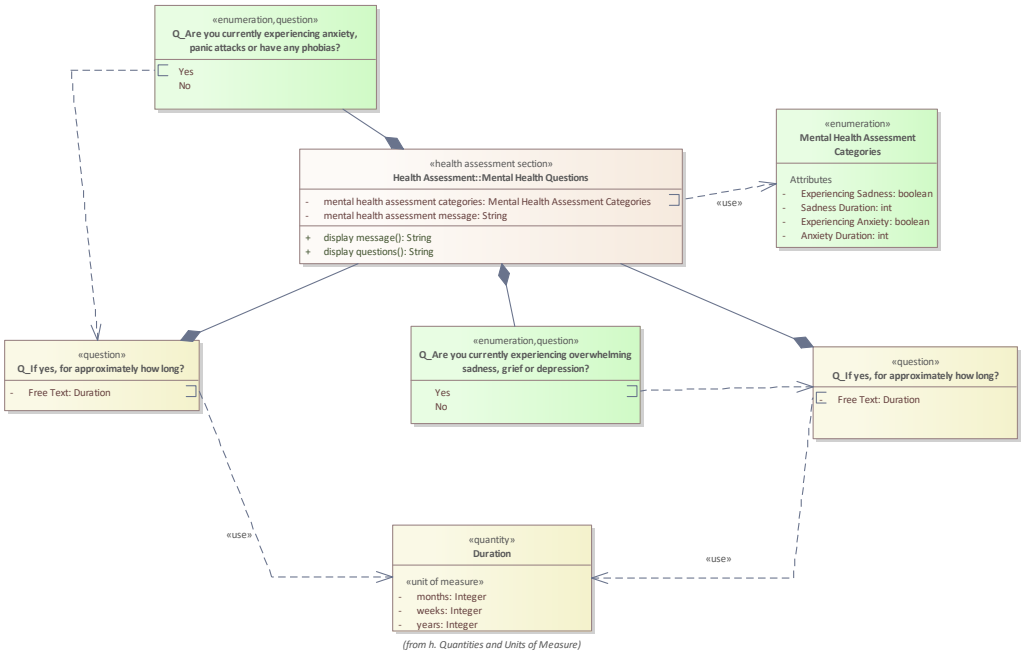


Figure 20: mental health questions

1.1.4.1.1.8 Blood Test Questions diagram

Class diagram in package 'Health Assessment'

Blood Test Questions
Version 1.0
Author: Moe Huss

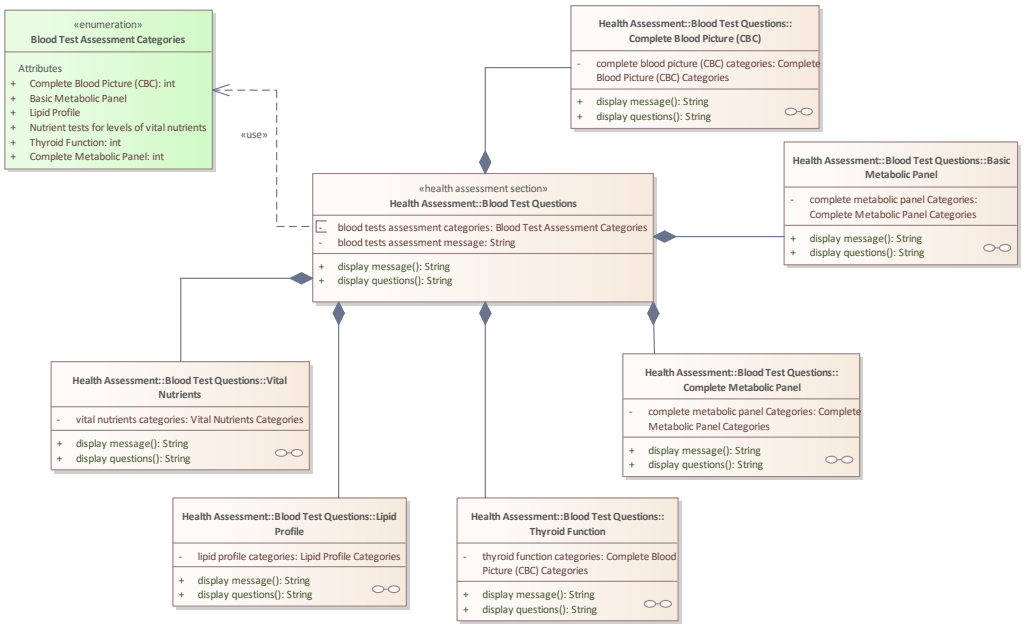


Figure 21: Blood Test Questions

1.1.4.1.1.8.1 Complete Blood Picture (CBC) diagram

Class diagram in package 'Health Assessment'

Complete Blood Picture (CBC)
Version 1.0
Author: Moe Huss

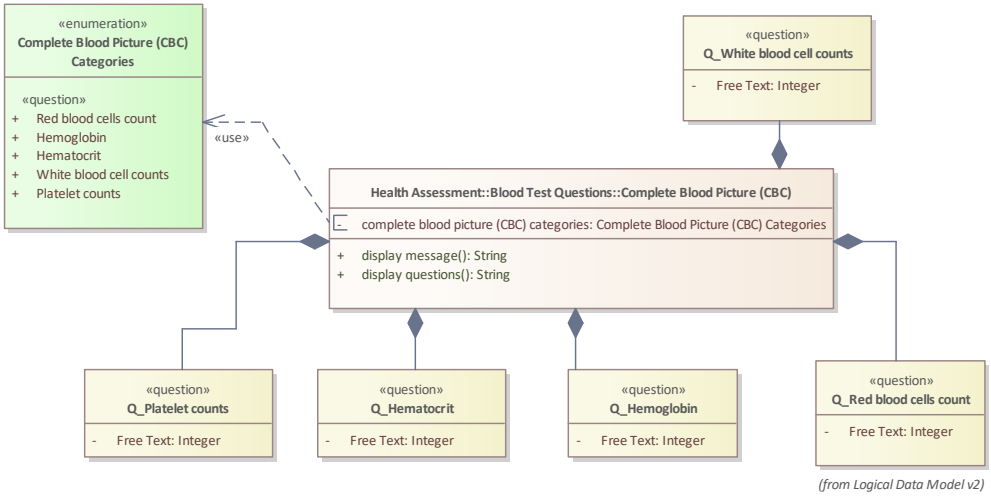


Figure 22: Complete Blood Picture (CBC)

1.1.4.1.1.8.2 Thyroid Function diagram

Class diagram in package 'Health Assessment'

Thyroid Function
Version 1.0
Author: Moe Huss

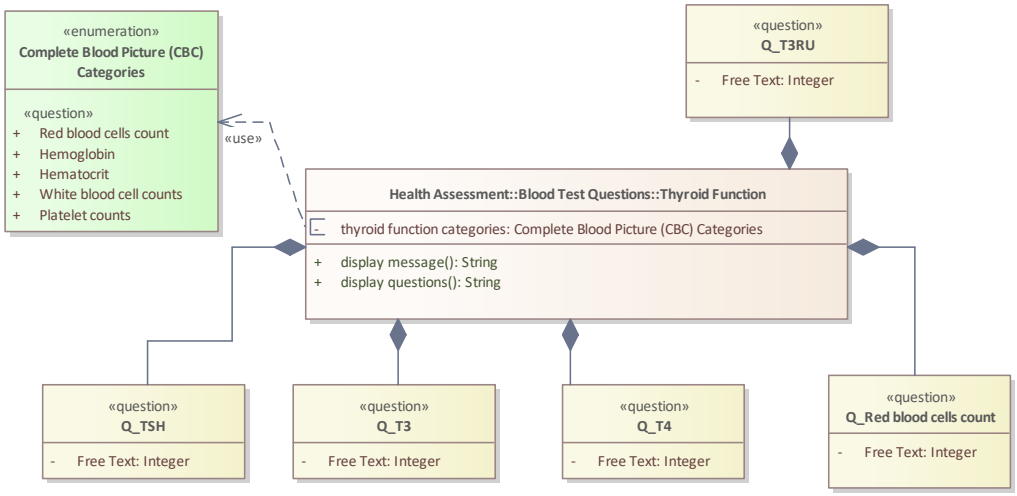


Figure 23: Thyroid Function

1.1.4.1.1.8.3 Complete Metabolic Panel diagram

Class diagram in package 'Health Assessment'

Complete Metabolic Panel
Version 1.0
Author: Moe Huss

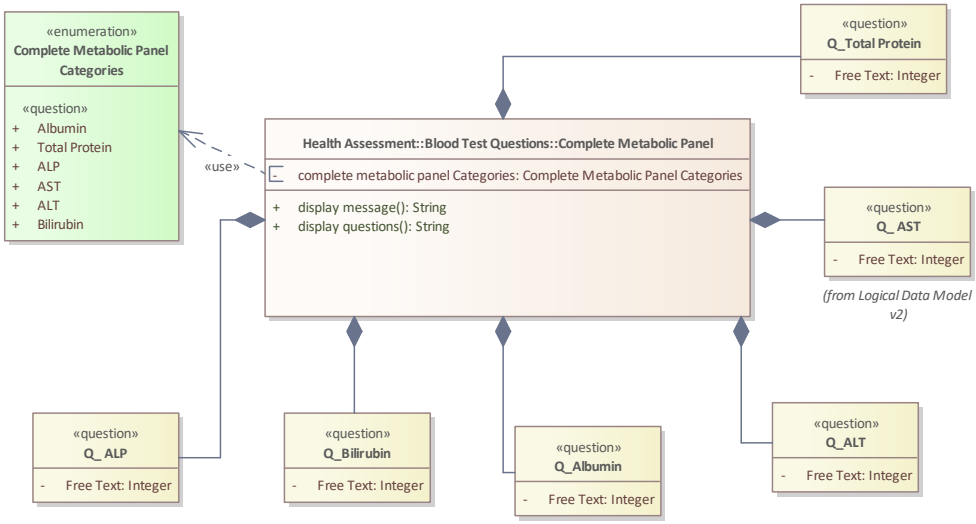


Figure 24: Complete Metabolic Panel

1.1.4.1.1.8.4 Lipid Profile diagram

Class diagram in package 'Health Assessment'

Lipid Profile
Version 1.0
Author: Moe Huss

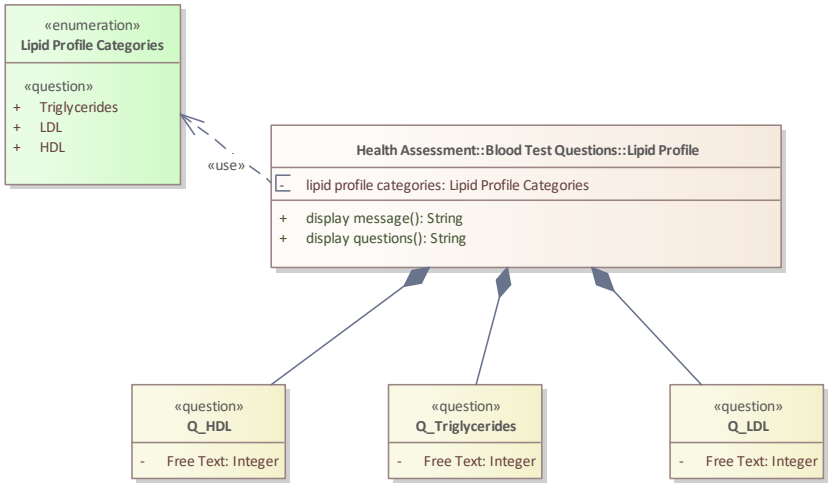


Figure 25: Lipid Profile

1.1.4.1.1.8.5 Vital Nutrients diagram

Class diagram in package 'Health Assessment'

Vital Nutrients
Version 1.0
Author: Moe Huss

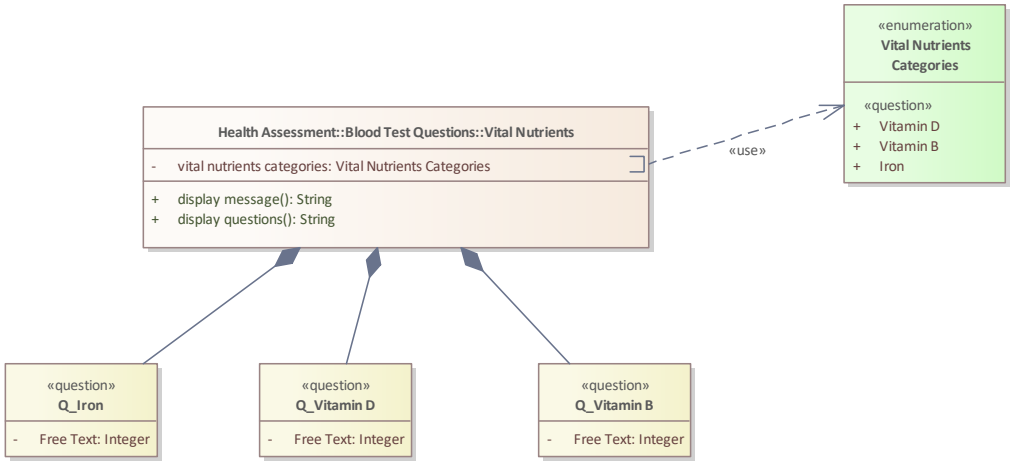


Figure 26: Vital Nutrients

1.1.4.1.1.8.6 Basic Metabolic Panel diagram

Class diagram in package 'Health Assessment'

Basic Metabolic Panel
Version 1.0
Author: Moe Huss

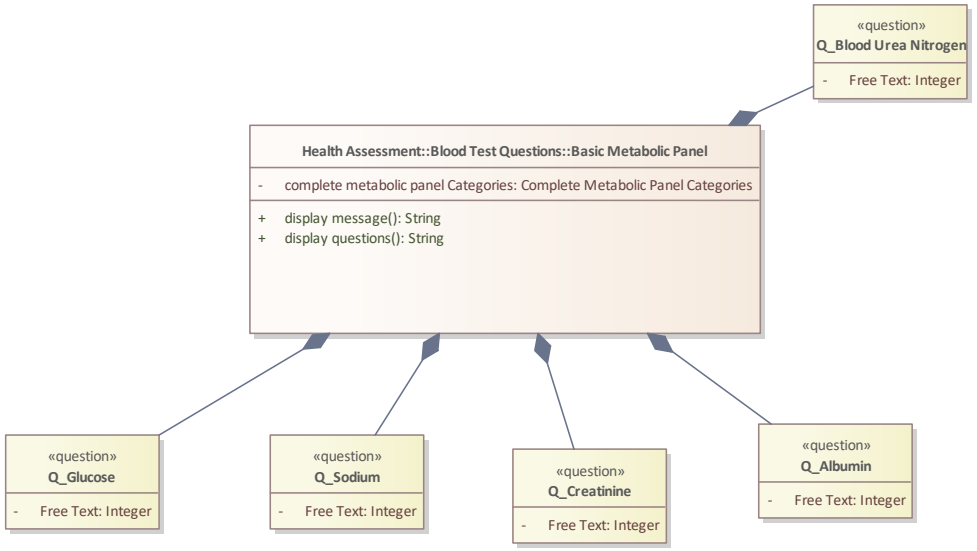


Figure 27: Basic Metabolic Panel

1.1.4.1.2 Health Report diagram

Class diagram in package 'Health Report'

Health Report
Version 1.0
Author: Moe Huss

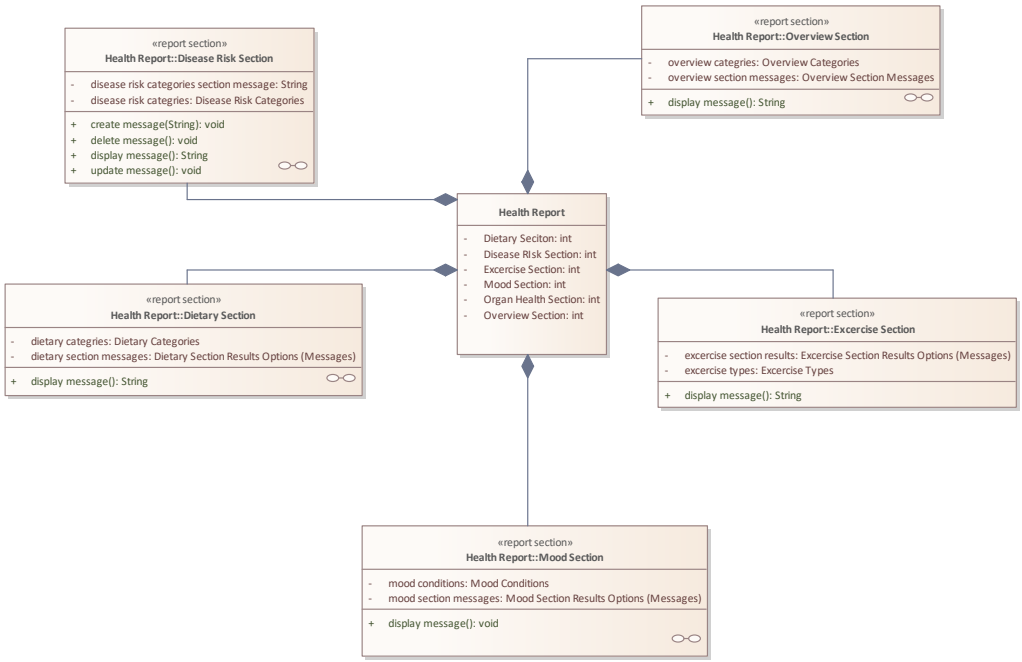


Figure 28: Health Report

1.1.4.1.2.1 Overview diagram

Class diagram in package 'Health Report'

Overview
Version 1.0
Author: Moe Huss

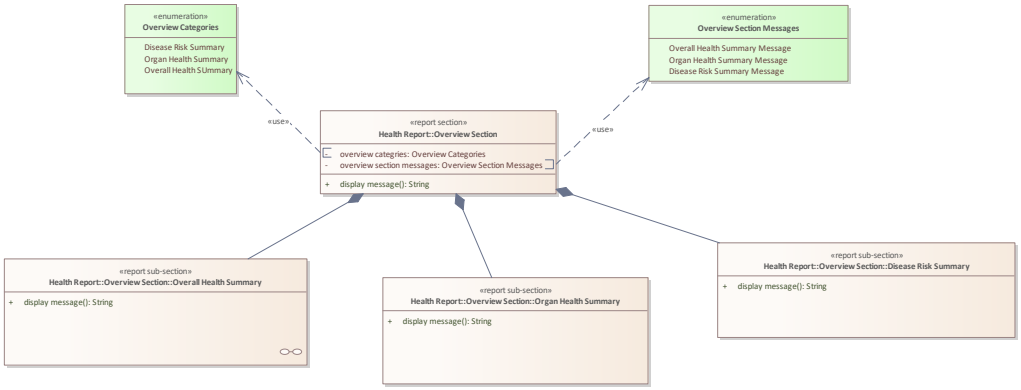


Figure 29: Overview

1.1.4.1.2.1.1 Overall Health Summary diagram

Class diagram in package 'Health Report'

Overall Health Summary
Version 1.0
Author: Moe Huss

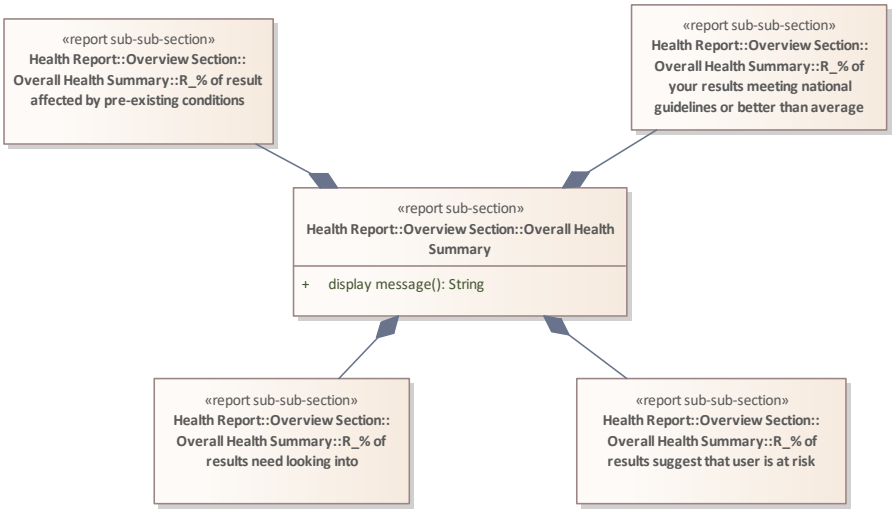


Figure 30: Overall Health Summary

1.1.4.1.2.1.2 Disease Risk Summary diagram

Class diagram in package 'Health Report'

Disease Risk Summary
Version 1.0
Author: Moe Huss

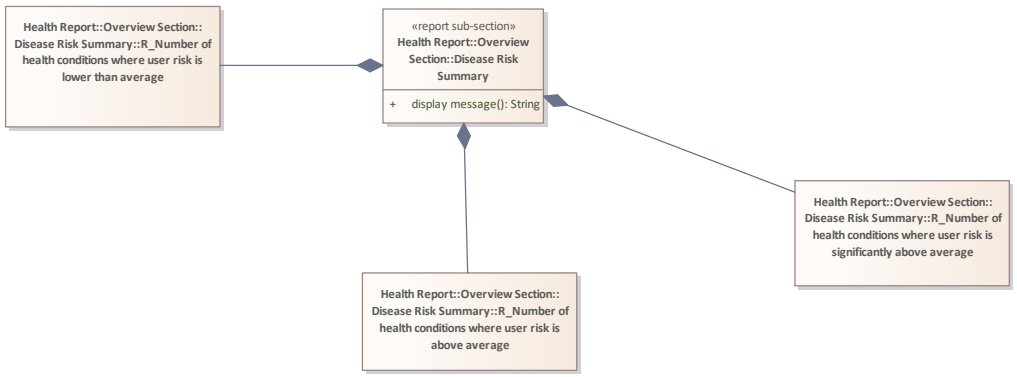


Figure 31: Disease Risk Summary

1.1.4.1.2.1.3 Organ Health Summary CDM diagram

Class diagram in package 'Health Report'

Organ Health Summary CDM
Version 1.0
Author: Moe Huss

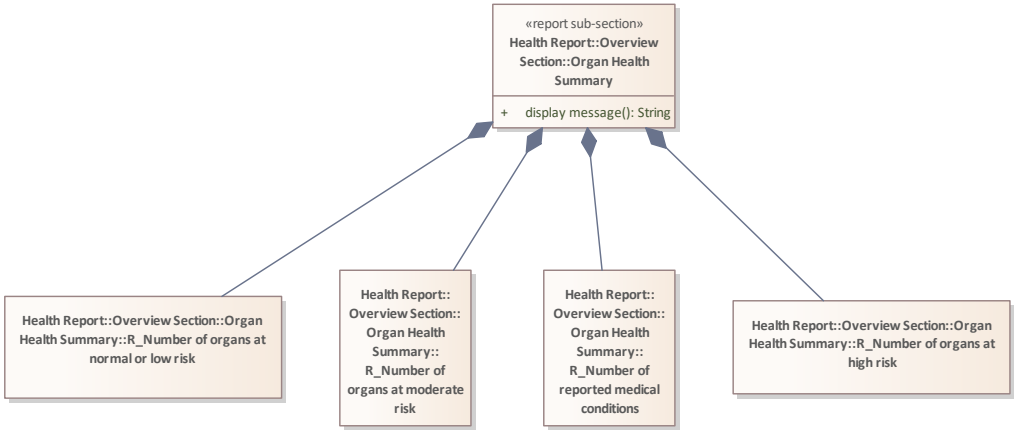


Figure 32: Organ Health Summary CDM

1.1.4.1.2.2 Disease Risk diagram

Class diagram in package 'Health Report'

Disease Risk
Version 1.0
Author: Moe Huss

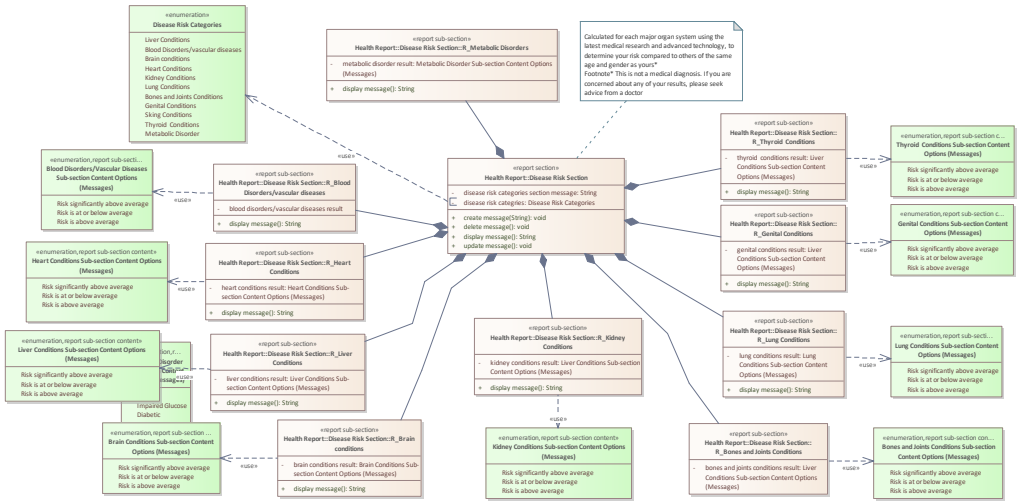


Figure 33: Disease Risk

1.1.4.1.2.3 Organ Health diagram

Class diagram in package 'Health Report'

Organ Health
Version 1.0
Author: Moe Huss

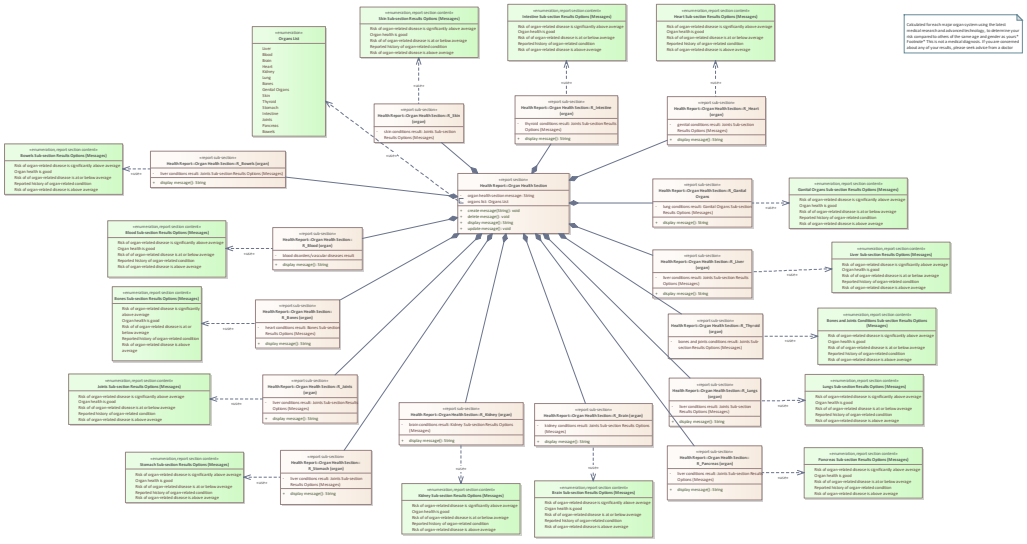


Figure 34: Organ Health

1.1.4.1.2.4 Dietary diagram

Class diagram in package 'Health Report'

Dietary
Version 1.0
Author: Moe Huss

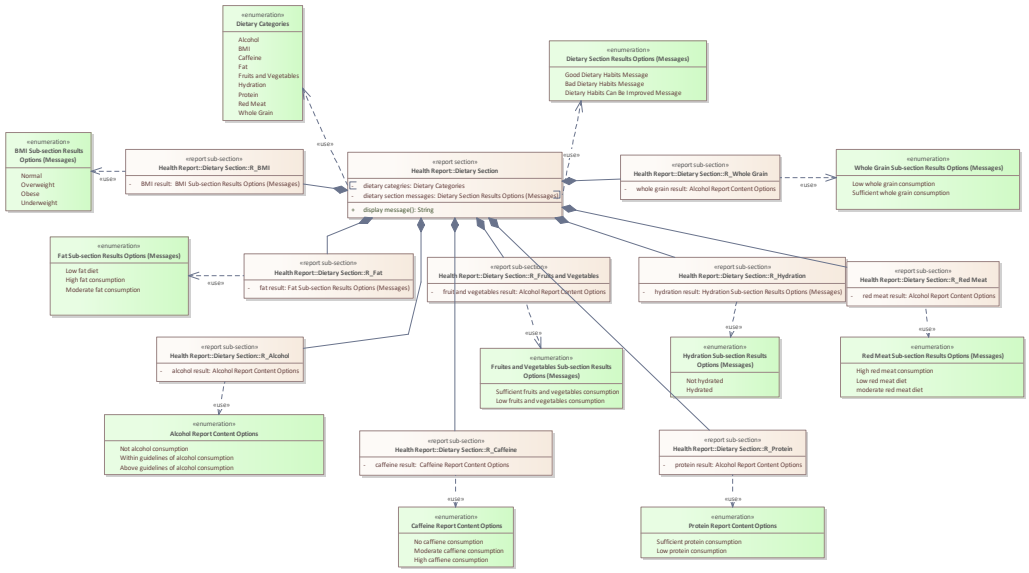


Figure 35: Dietary

1.1.4.1.2.5 Mood diagram

Class diagram in package 'Health Report'

Mood
Version 1.0
Author: Moe Huss

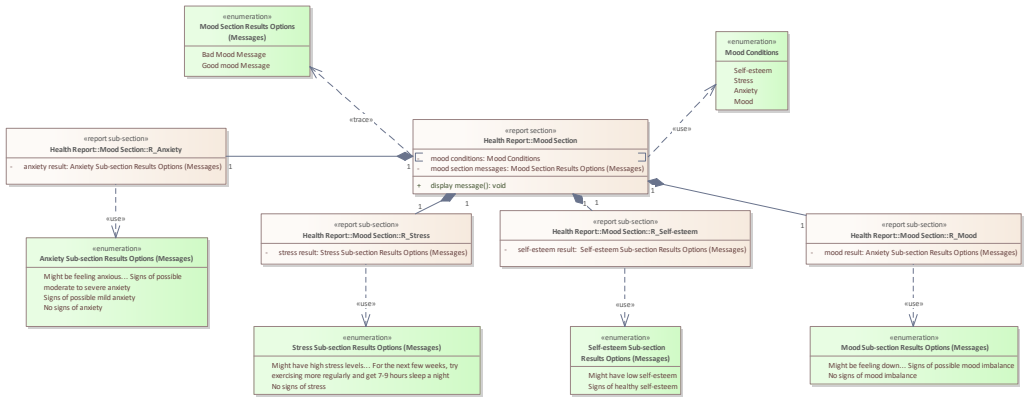


Figure 36: Mood

1.1.5 Physical Data Model 2 diagram

Class diagram in package 'Physical Data Model v2'

Physical Data Model 2
Version
Author: Moe Huss

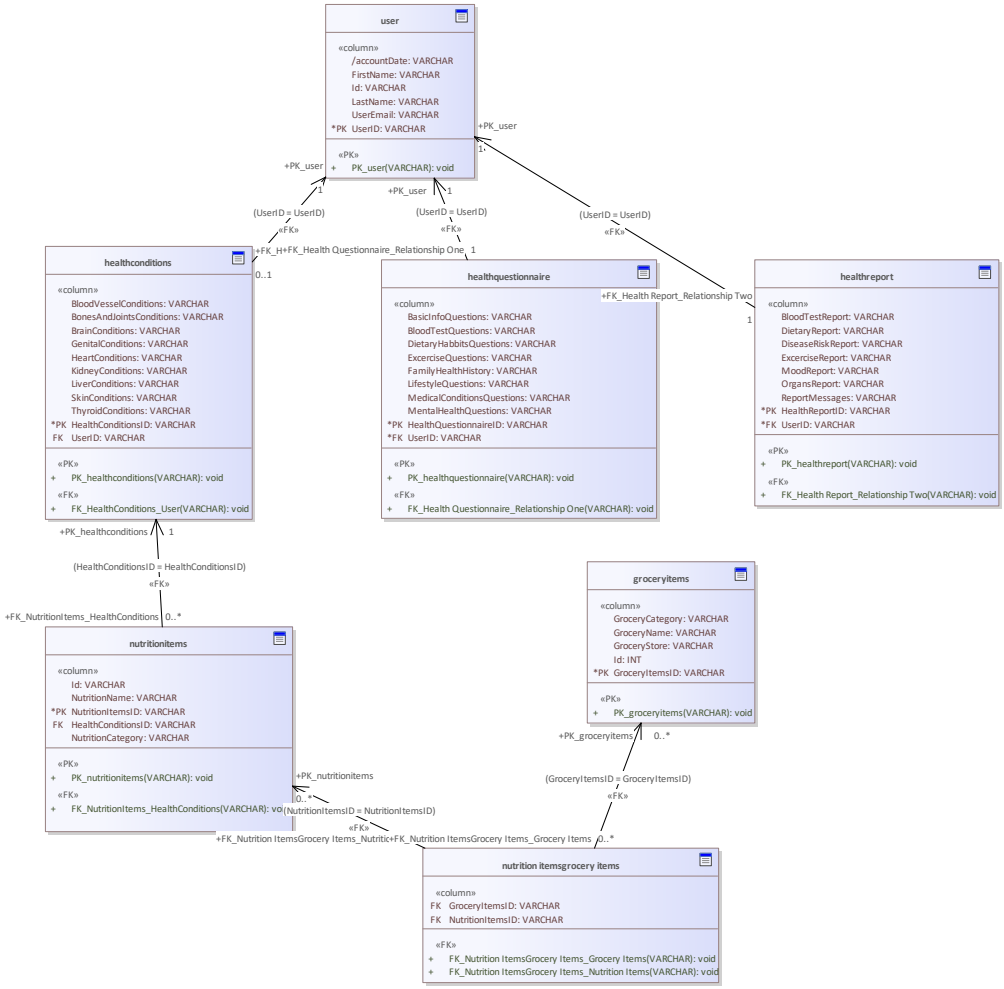


Figure 37: Physical Data Model 2

1.1.6 Logical Data Model v3 diagram

Class diagram in package 'Logical Data Model v3'

Logical Data Model v3
Version 1.0
Author: Moe Huss

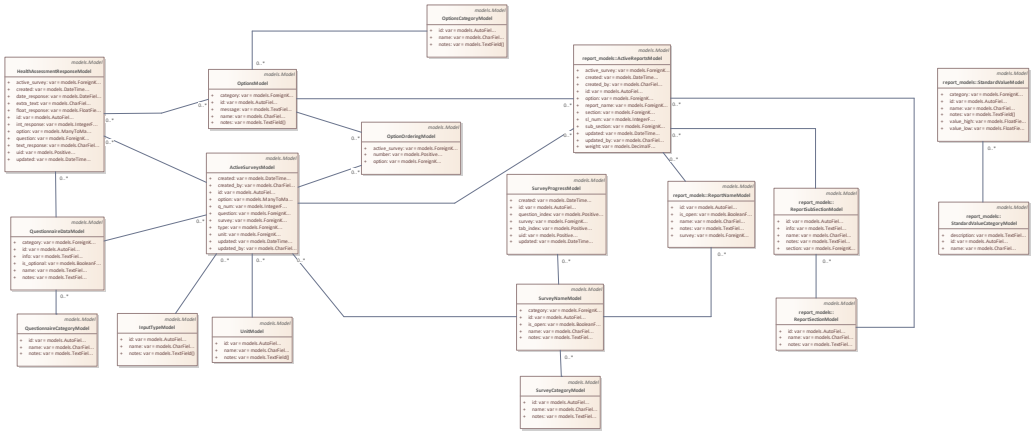


Figure 38: Logical Data Model v3

1.1.6.1 PDM Diagram diagram

Data Modeling diagram in package 'Physical Data Model v3'

PDM Diagram
Version 1.0
Author: Moe Huss

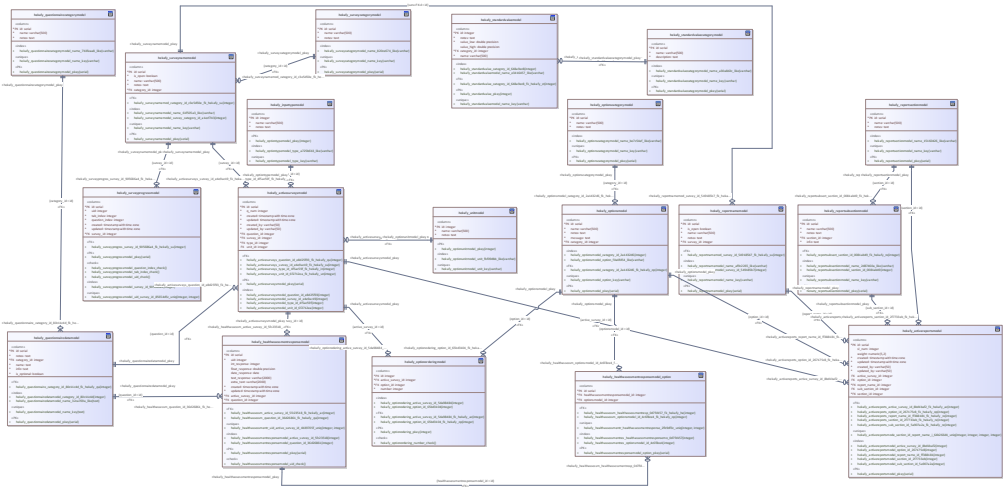


Figure 39: PDM Diagram

Diagram Report

e. Context

Version 1.0 • Final

Date Generated: 2/21/2022

Author: Moe Huss

CREATED WITH ENTERPRISE ARCHITECT

Table of Contents

1.1 Alpha v2 value stream diagram

Use Case diagram in package 'e. Context'

Alpha v2 value stream
Version 1.0
Author: Moe Huss

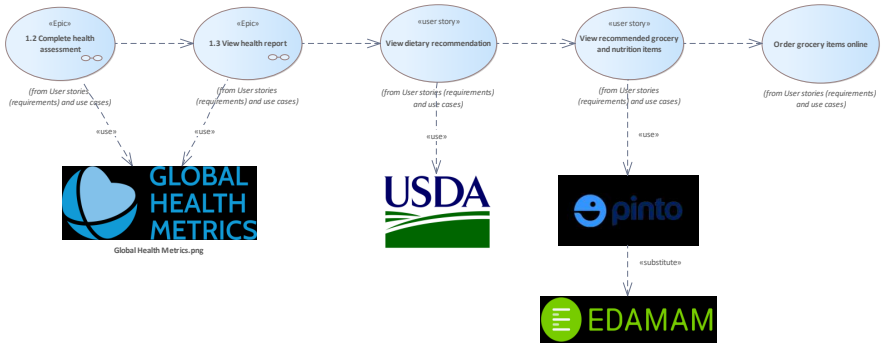


Figure 1: Alpha v2 value stream

1.2 hekafy hospital analogy diagram

Class diagram in package 'e. Context'

hekafy hospital analogy
Version 1.0
Author: Moe Huss

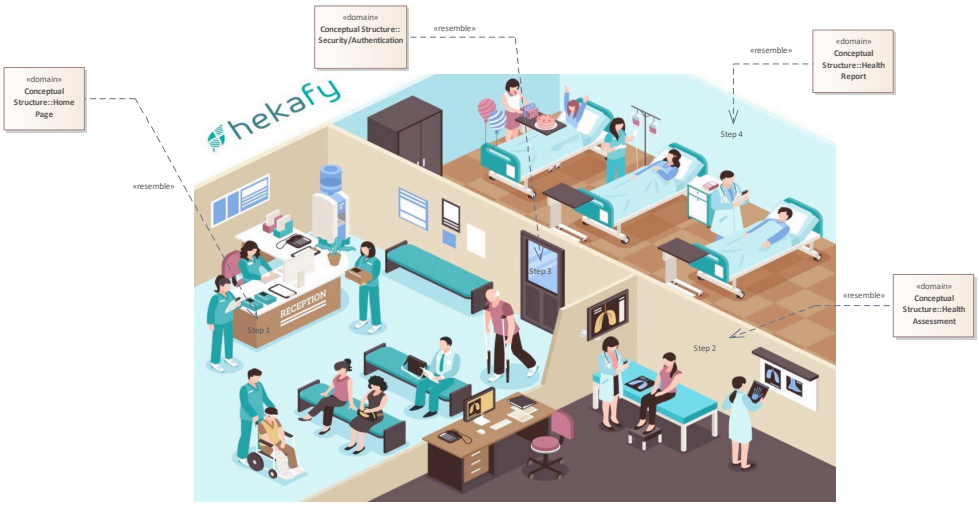


Figure 2: hekafy hospital analogy

1.3 Hekafy Value Stream diagram

Class diagram in package 'e. Context'

Hekafy Value Stream
Version 1.0
Author: Moe Huss

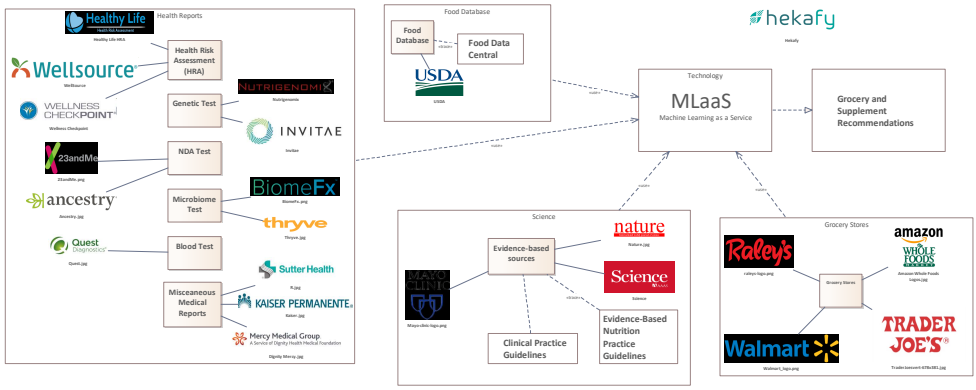


Figure 3: Hekafy Value Stream

1.4 hekafy Value Stream_Raleys diagram

Class diagram in package 'e. Context'

Hekafy Value Stream_Raleys
Version 1.0
Author: Moe Huss

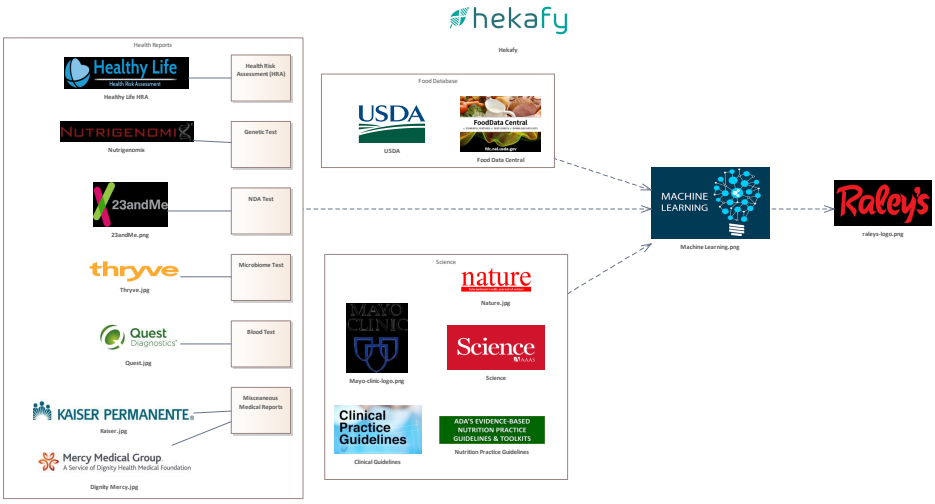


Figure 4: hekafy Value Stream_Raleys

1.4.1 Business Model Canvas diagram

Class diagram in package 'Business Model Canvas'

This Class diagram uses a series of boundaries to create the structure of the canvas. Classes can be added to each section of the canvas as needed.

Business Model Canvas
Version 1.0
Author: Moe Huss

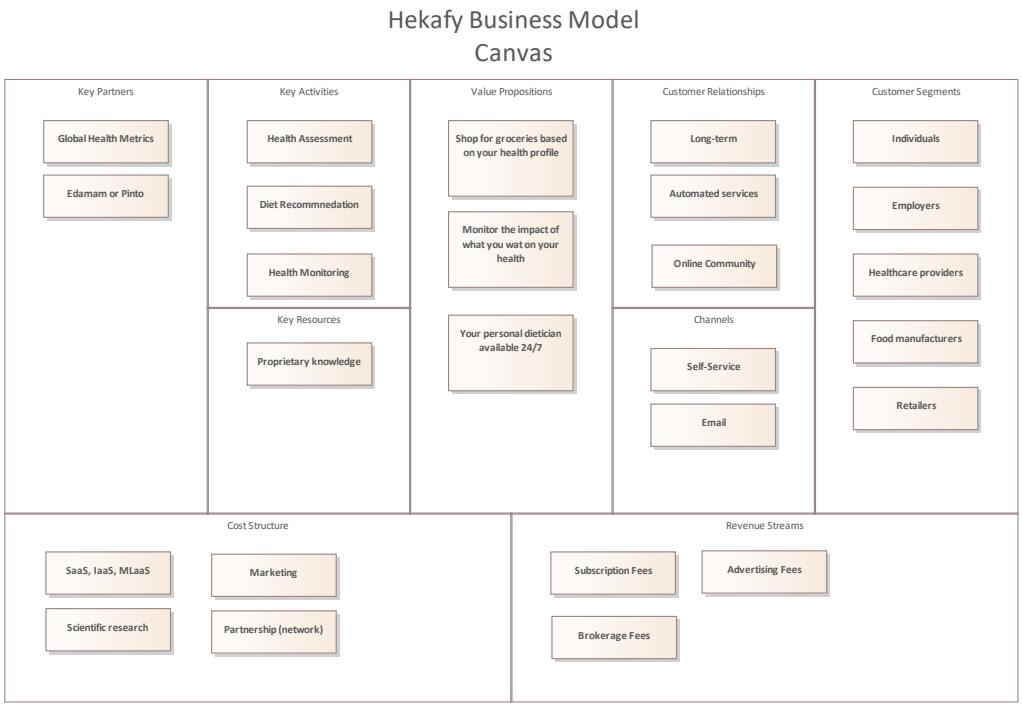


Figure 5: Business Model Canvas

1.4.2 Mind Mapping Diagram diagram

Mind Mapping diagram in package 'Success factors of Agile SW projects'

Mind Mapping Diagram
Version 1.0
Author: Moe Huss

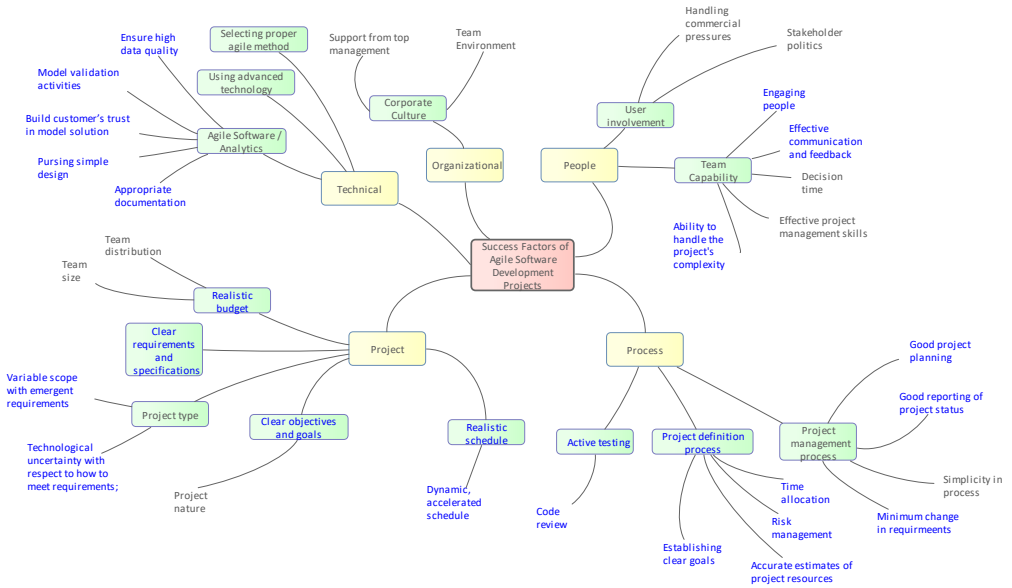


Figure 6: Mind Mapping Diagram

1.4.3 Main research tasks_Tasks 1-15 diagram

Activity diagram in package 'Main research tasks'

Main research tasks_Tasks 1-15
Version 1.0
Author: Moe Huss

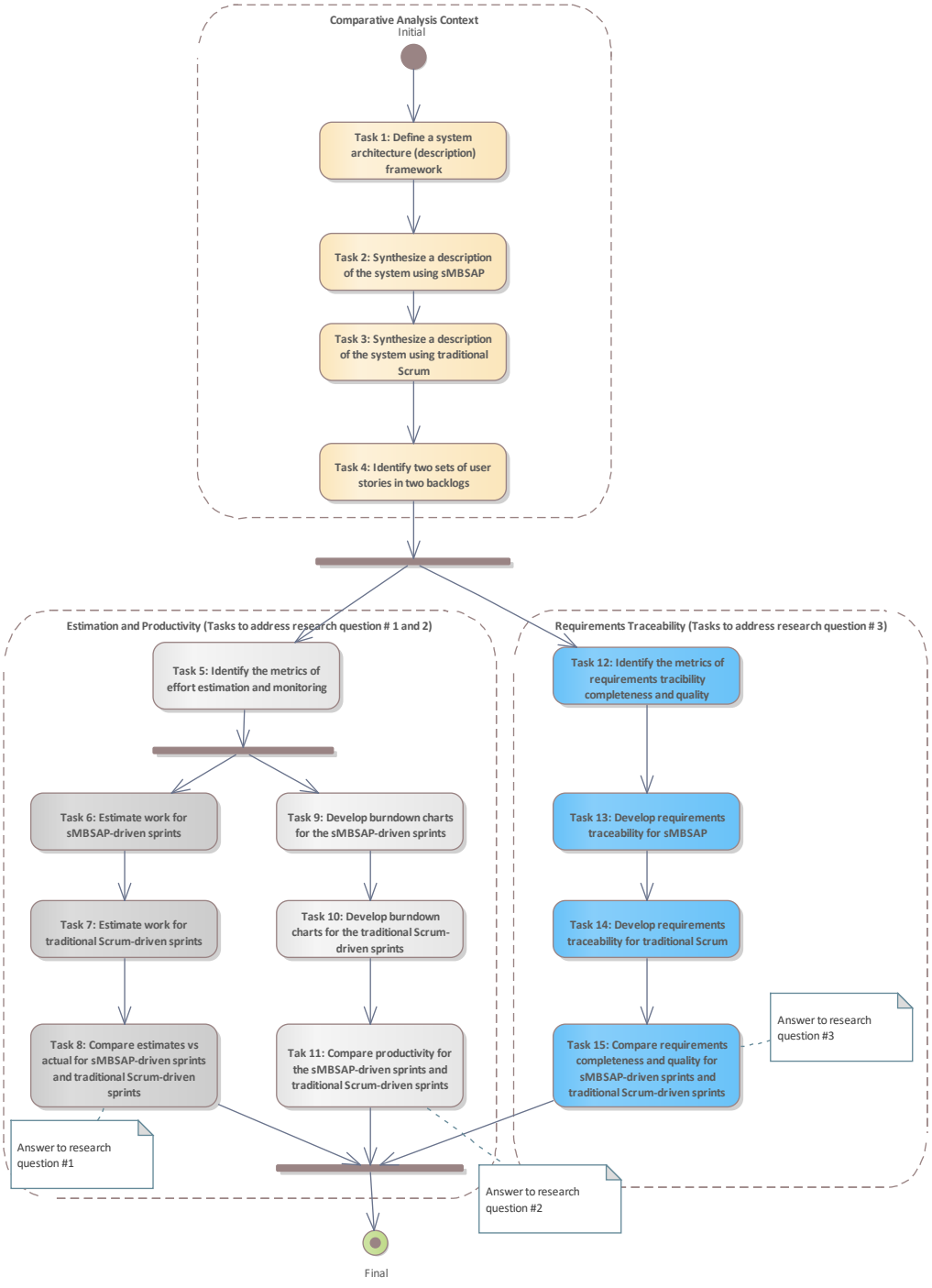


Figure 7: Main research tasks_Tasks 1-15

Diagram Report

f. System Screens Wireframe

Version 1.0 • Final

Date Generated: 2/21/2022

Author: Moe Huss

CREATED WITH ENTERPRISE ARCHITECT

Table of Contents

1.1 Registration screen diagram

Dialog Wireframe diagram in package 'f. System Screens Wireframe'

Registration screen
Version 1.0
Author: Moe Huss

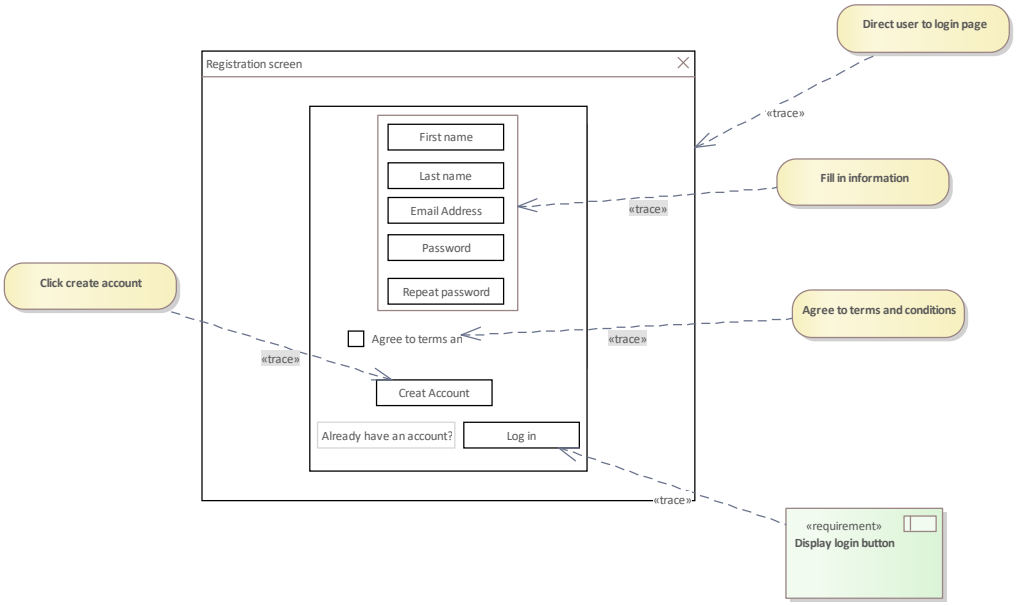


Figure 1: Registration screen

1.2 Login screen wireframe diagram

Dialog Wireframe diagram in package 'f. System Screens Wireframe'

Login screen wireframe
Version 1.0
Author: Moe Huss

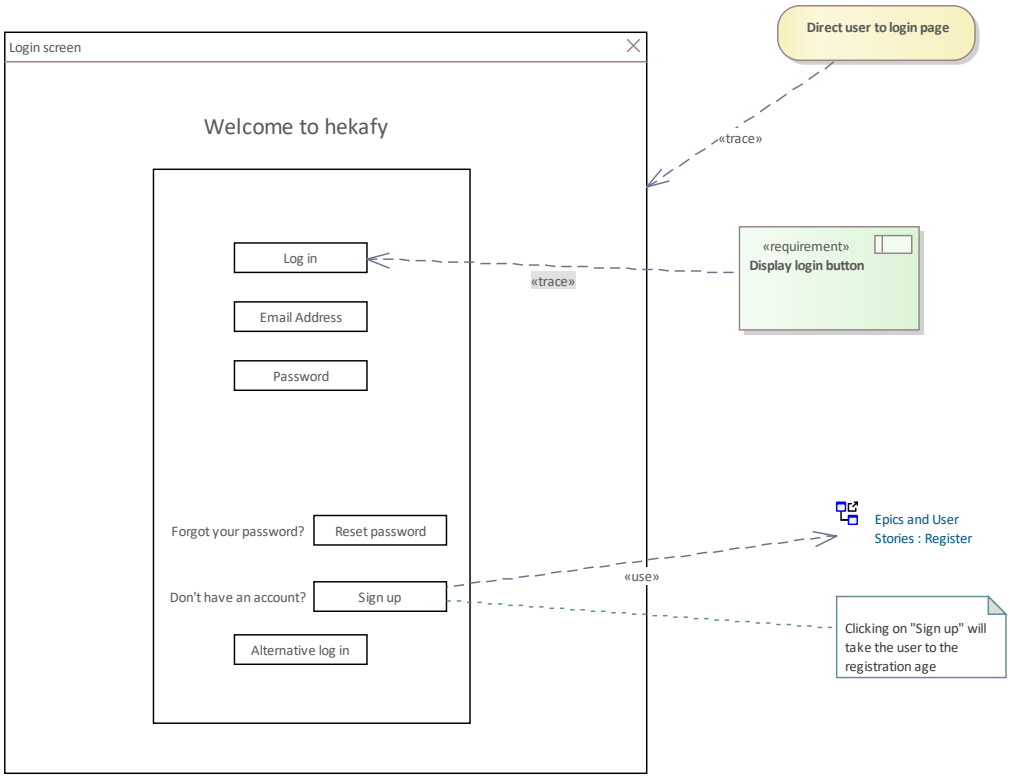


Figure 2: Login screen wireframe

1.3 Email verification page diagram

Dialog Wireframe diagram in package 'f. System Screens Wireframe'

Email verification page
Version 1.0
Author: Moe Huss

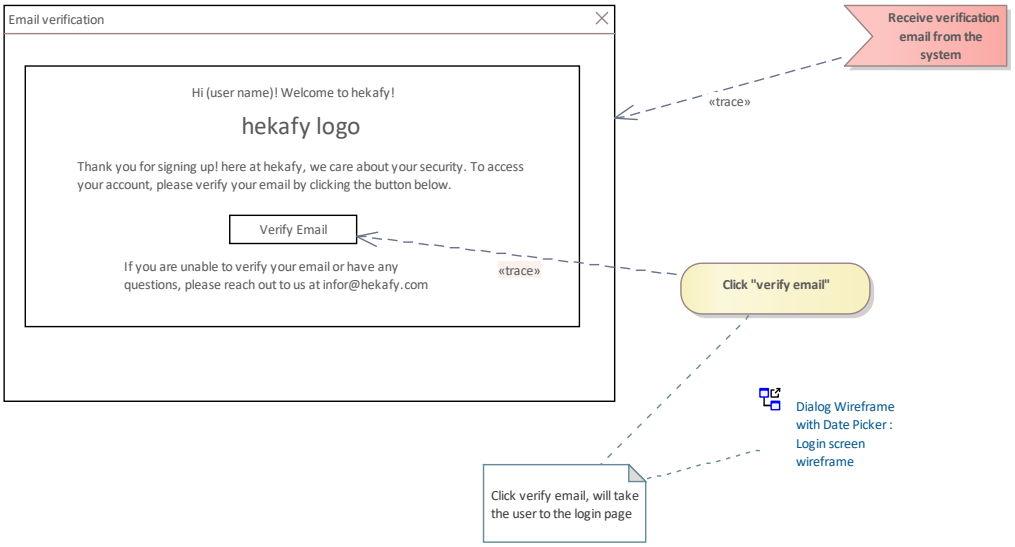


Figure 3: Email verification page

1.4 Health assessment welcome screen diagram

Dialog Wireframe diagram in package 'f. System Screens Wireframe'

Health assessmnet welcome screen
Version 1.0
Author: Moe Huss

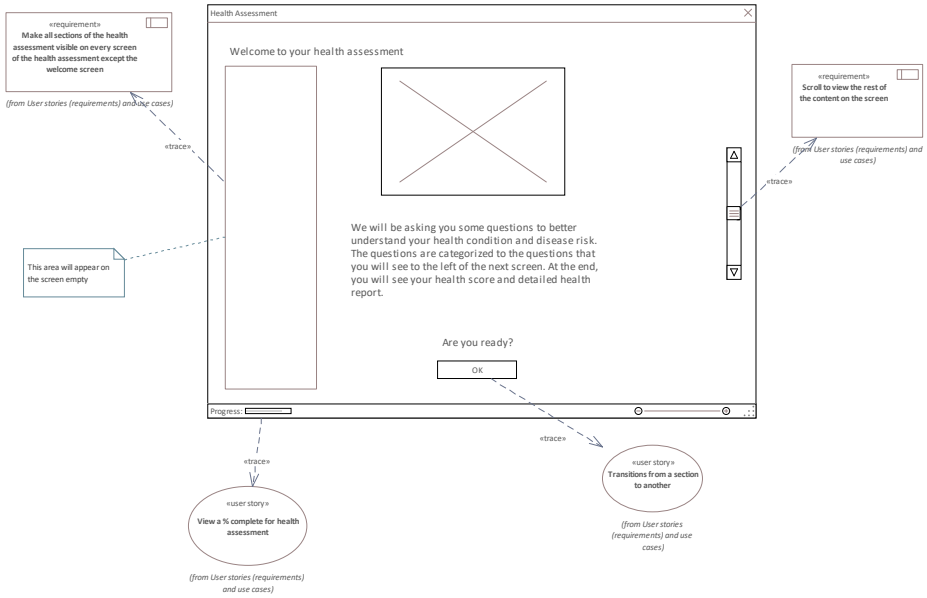


Figure 4: Health assessmnet welcome screen

1.5 Health Assessment Question Screen diagram

Dialog Wireframe diagram in package 'f. System Screens Wireframe'

Health Assessment Question Screen
Version 1.0
Author: Moe Huss

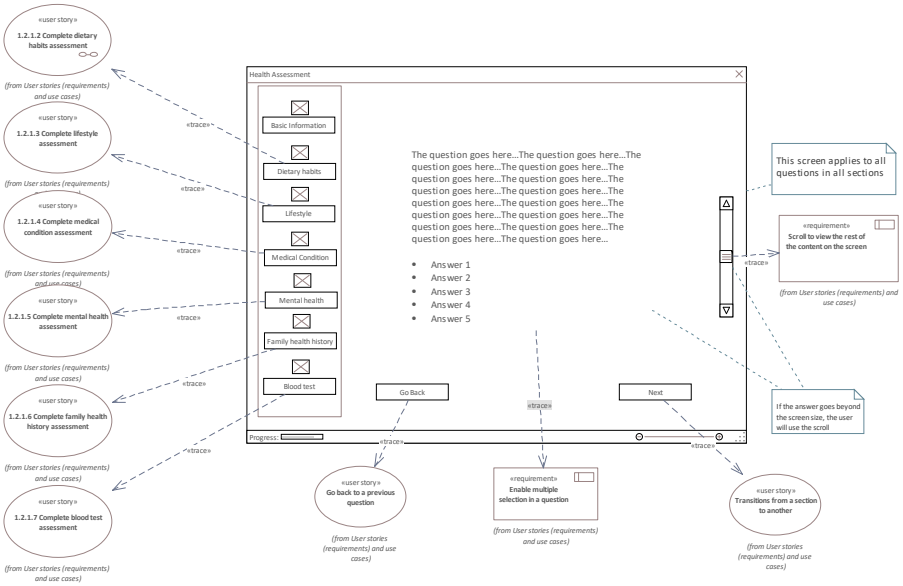


Figure 5: Health Assessmnet Question Screen

1.6 Health Assessment transition: info to diet habits diagram

Dialog Wireframe diagram in package 'f. System Screens Wireframe'

Health Assessment transition: info to diet habits
Version 1.0
Author: Moe Huss

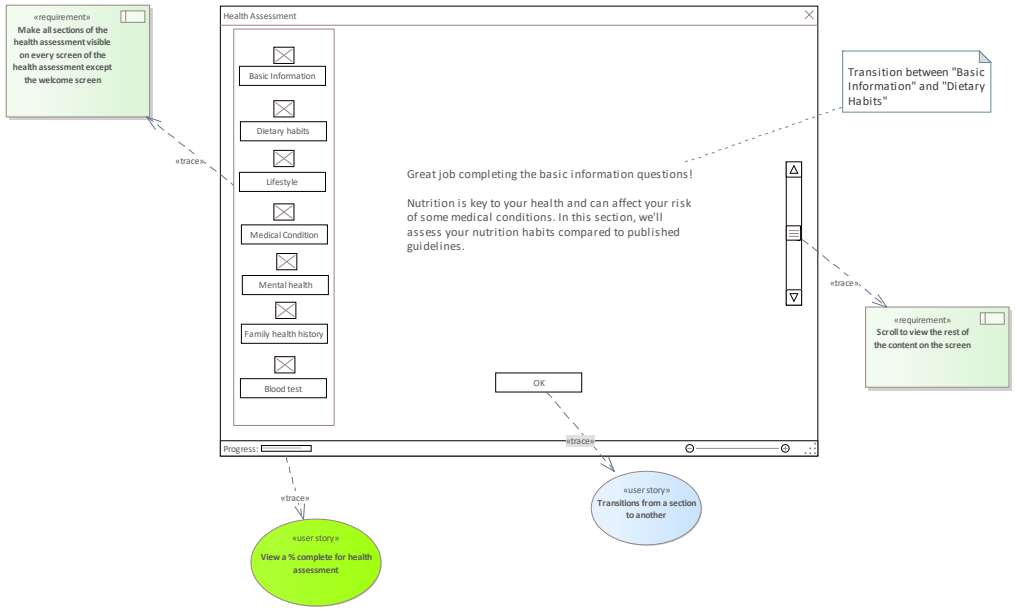


Figure 6: Health Assessment transition: info to diet habits

1.7 Health Assessment transition : dietary habits to life style diagram

Dialog Wireframe diagram in package 'f. System Screens Wireframe'

Health Assessment transition : dietary habits to life style
Version 1.0
Author: Moe Huss

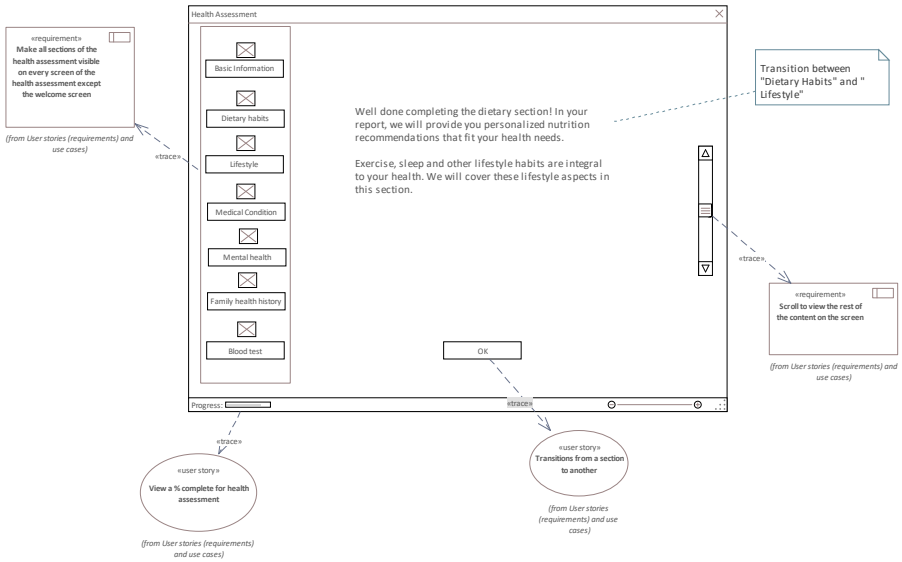


Figure 7: Health Assessment transition : dietary habits to life style

1.8 Health assessment transition: lifestyle to medical condition diagram

Dialog Wireframe diagram in package 'f. System Screens Wireframe'

Health assessment transition: lifestyle to medical condition
Version 1.0
Author: Moe Huss

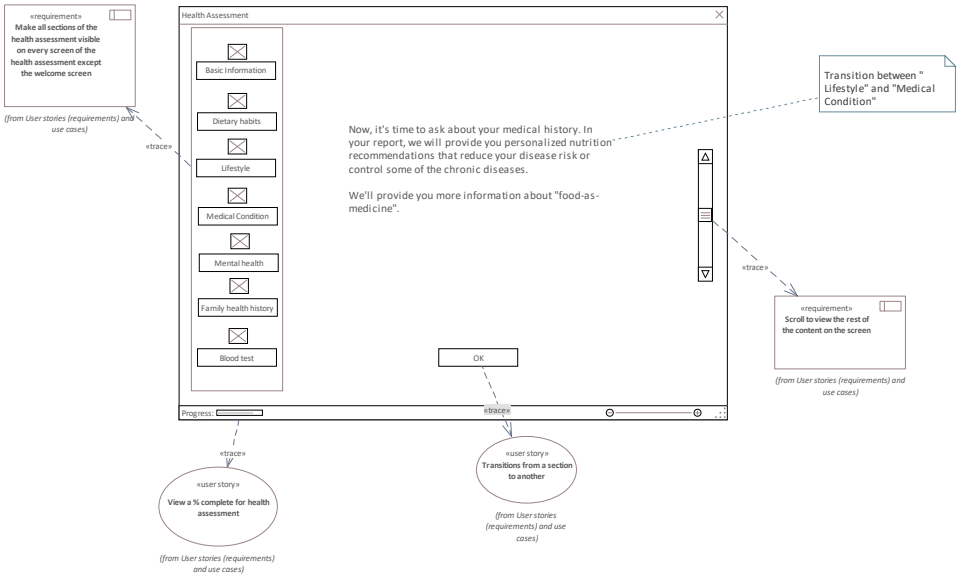


Figure 8: Health assessment transition: lifestyle to medical condition

1.9 Health assessment transition: medical condition to mental health diagram

Dialog Wireframe diagram in package 'f. System Screens Wireframe'

Health assessment transition: medical condition to mental health
Version 1.0
Author: Moe Huss

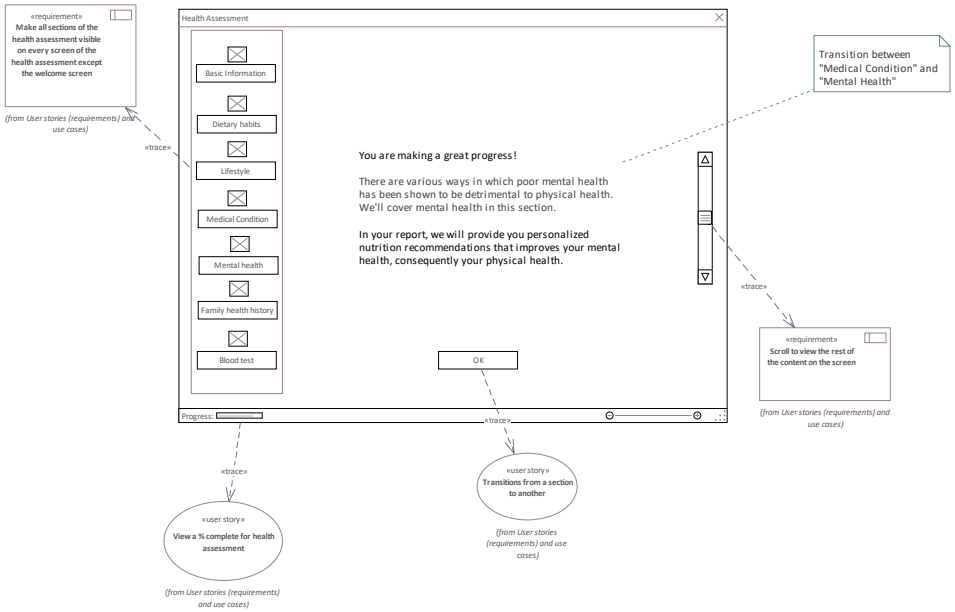


Figure 9: Health assessment transition: medical condition to mental health

1.10 Health Assessment transition: mental health to family history diagram

Dialog Wireframe diagram in package 'f. System Screens Wireframe'

Health Assessment transition: mental health to family history
Version 1.0
Author: Moe Huss

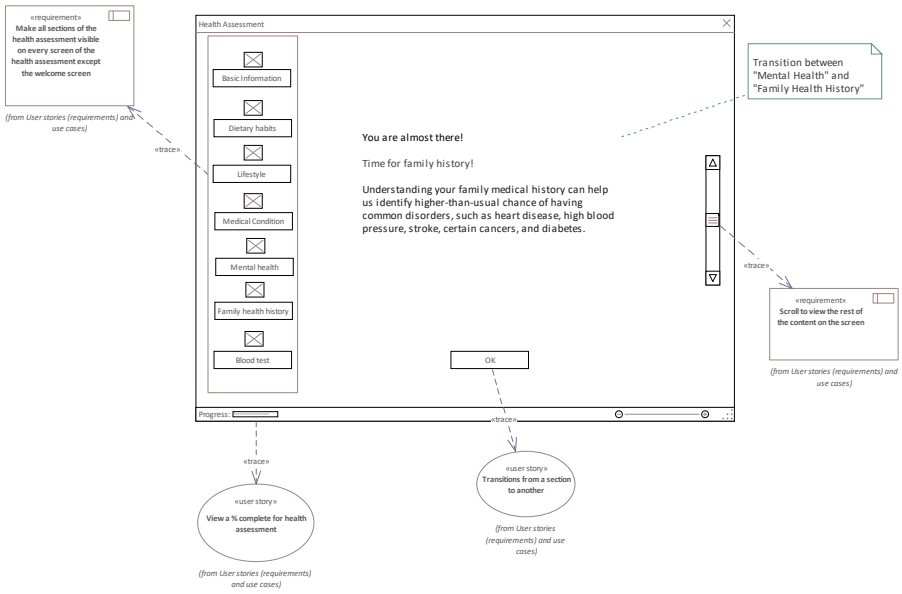


Figure 10: Health Assessment transition: mental health to family history

1.11 Health assessment: transition after family history diagram

Dialog Wireframe diagram in package 'f. System Screens Wireframe'

Health assessment: transition after family history
Version 1.0
Author: Moe Huss

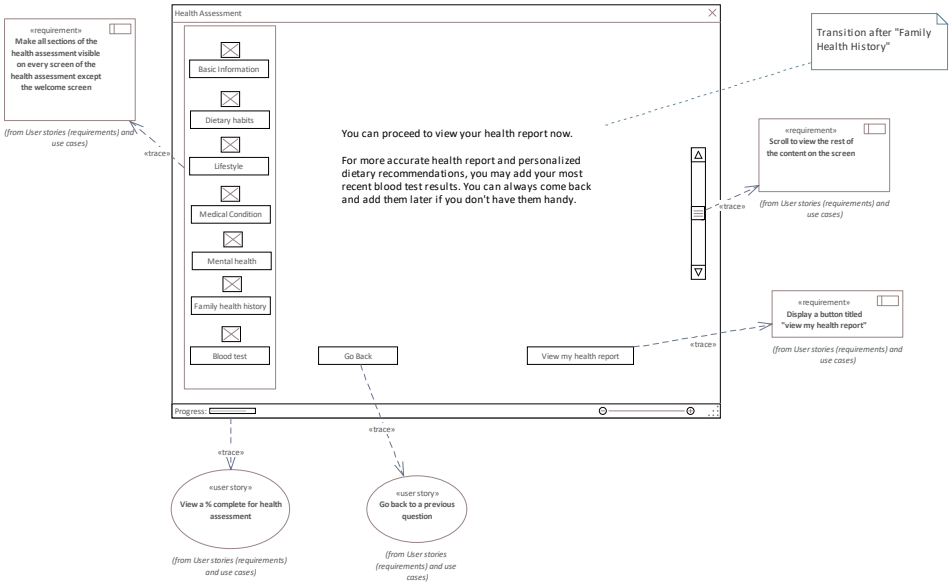


Figure 11: Health assessment: transition after family history

1.12 Health score diagram

Dialog Wireframe diagram in package 'f. System Screens Wireframe'

Health score
Version 1.0
Author: Moe Huss

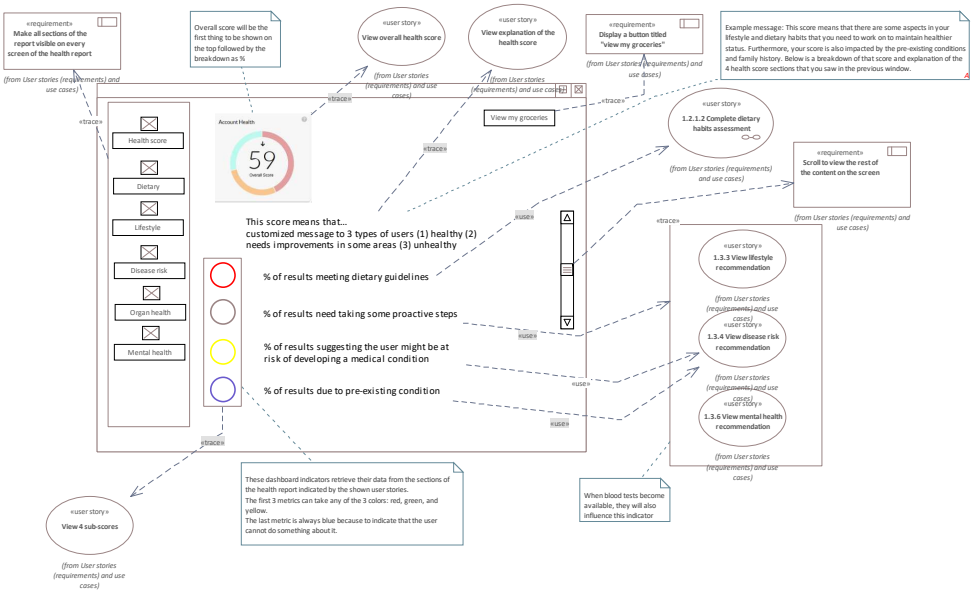


Figure 12: Health score

1.13 Dietary Screen diagram

Dialog Wireframe diagram in package 'f. System Screens Wireframe'

Dietary Screen
Version 1.0
Author: Moe Huss

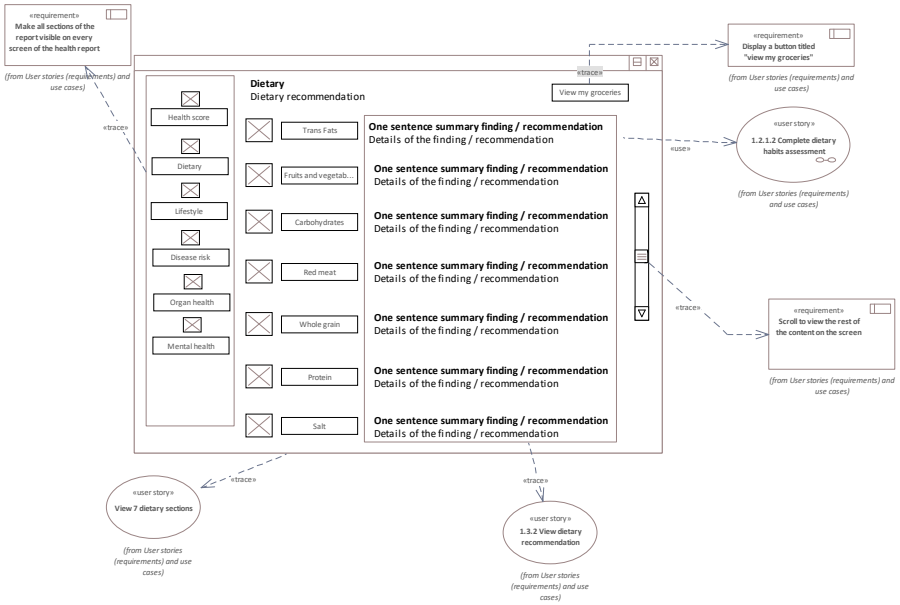


Figure 13: Dietary Screen

1.14 Lifestyle screen diagram

Dialog Wireframe diagram in package 'f. System Screens Wireframe'

Lifestyle screen
Version 1.0
Author: Moe Huss

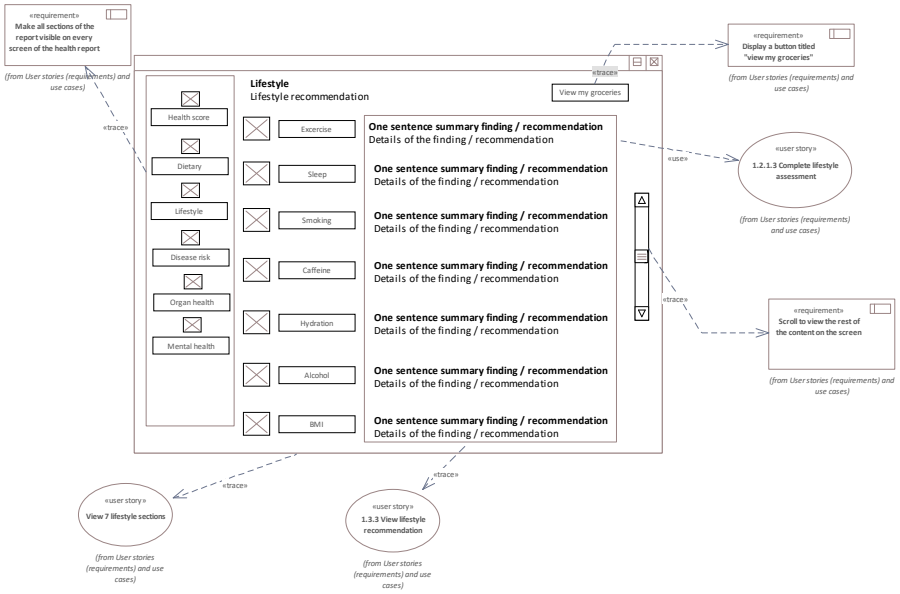


Figure 14: Lifestyle screen

1.15 Disease Risk screen diagram

Dialog Wireframe diagram in package 'f. System Screens Wireframe'

Disease Risk screen
Version 1.0
Author: Moe Huss

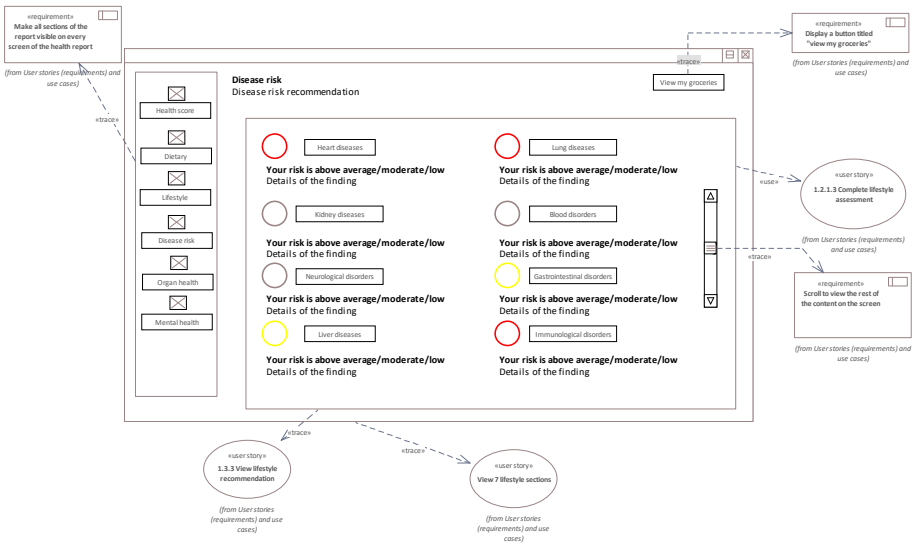


Figure 15: Disease Risk screen

1.16 Organ health screen diagram

Dialog Wireframe diagram in package 'f. System Screens Wireframe'

Organ health screen
Version 1.0
Author: Moe Huss

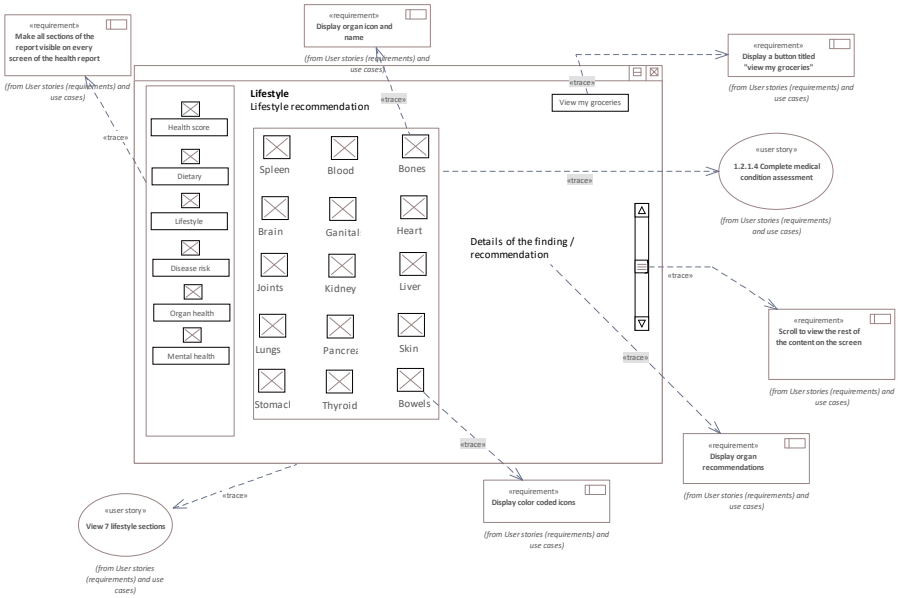


Figure 16: Organ health screen

1.17 Mental health screen diagram

Dialog Wireframe diagram in package 'f. System Screens Wireframe'

Mental health screen
Version 1.0
Author: Moe Huss

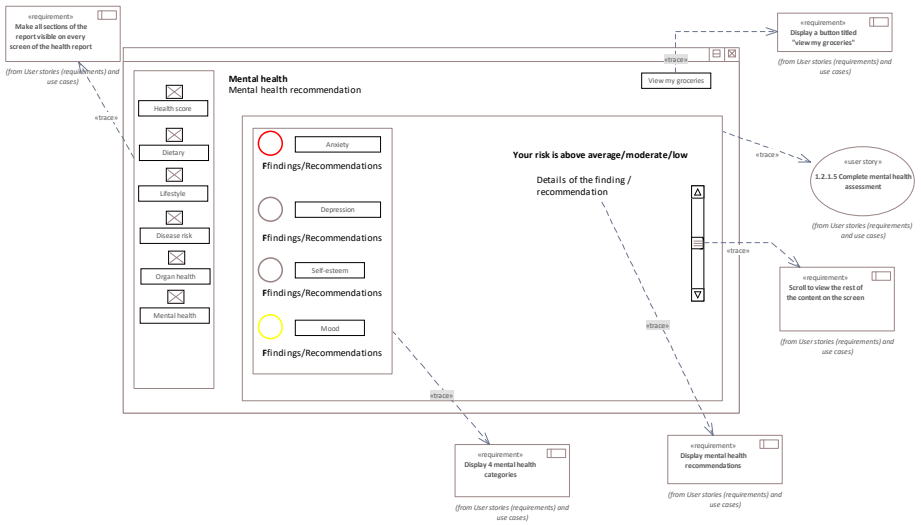


Figure 17: Mental health screen

1.17.1 Webpage Wireframe with Calendar diagram

Webpage Wireframe diagram in package 'Hekafy Webpage Wireframe'

Webpage Wireframe with Calendar
Version 1.0
Author: Moe Huss

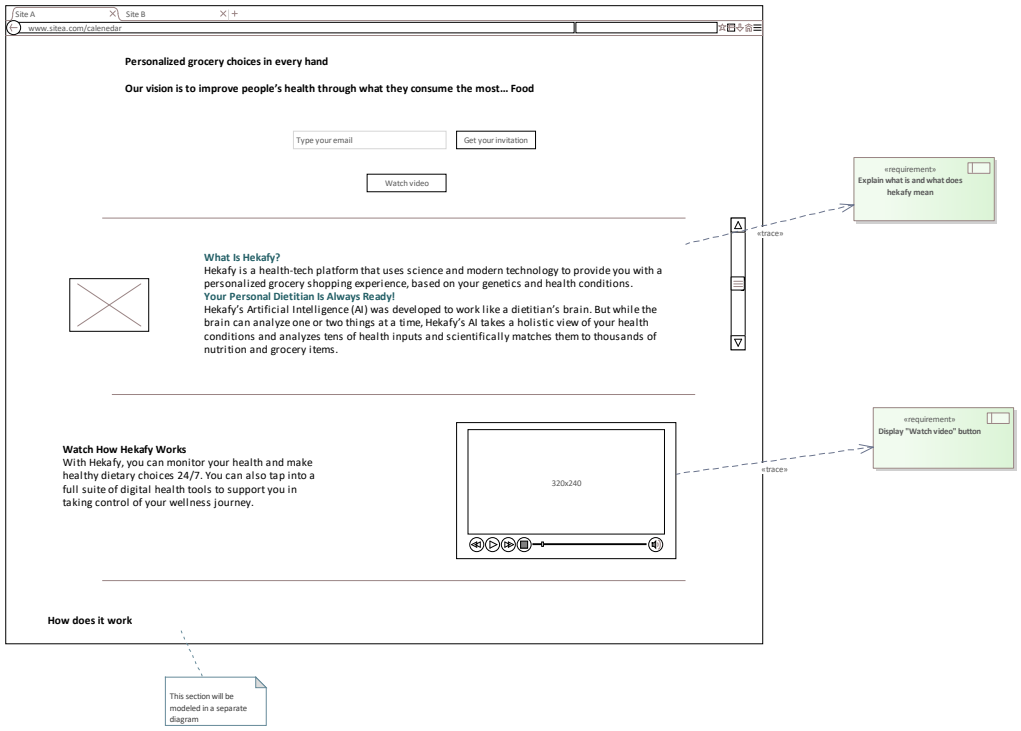


Figure 18: Webpage Wireframe with Calendar

1.17.2 Hekafy Webpage Wireframe diagram

Webpage Wireframe diagram in package 'Hekafy Webpage Wireframe'

Hekafy Webpage Wireframe
Version 1.0
Author: Moe Huss

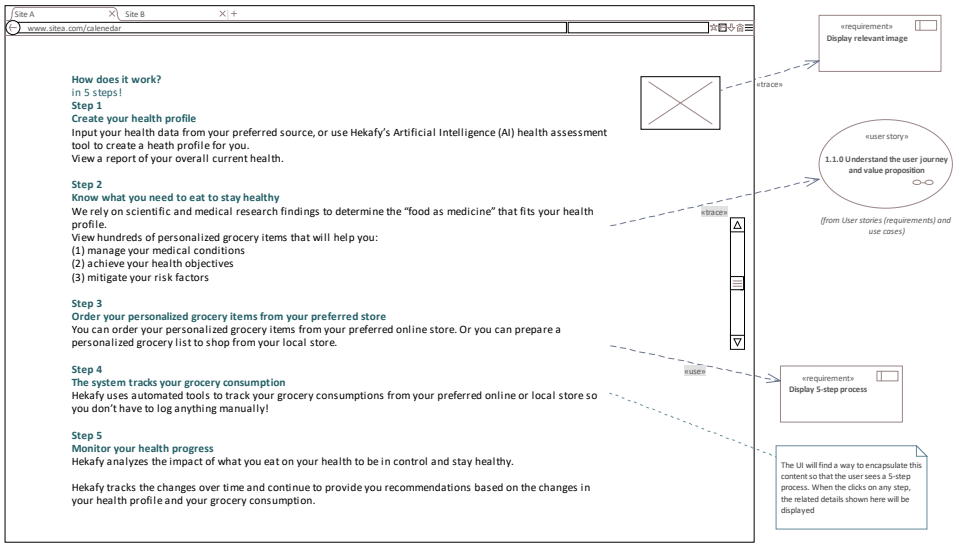


Figure 19: Hekafy Webpage Wireframe

Diagram Report

g. Decision Model

Version 1.0 • Final

Date Generated: 2/21/2022

Author: Moe Huss

CREATED WITH ENTERPRISE ARCHITECT

Table of Contents

1.1.1 Decision With BKM diagram

Decision Requirements diagram in package 'Decision With BKM'

Decision With BKM
Version 1.0
Author: Moe Huss

Decision model using the Decision Model Notation (DMN) to simulate deciding the hydration score based on the user responses

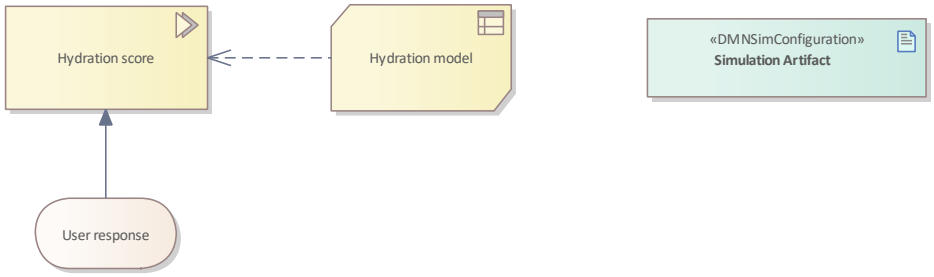


Figure 1: Decision With BKM

1.1.2 DecisionTable diagram

Decision Requirements diagram in package 'Blood Oxygen Saturation'

DecisionTable
Version 1.0
Author: Moe Huss

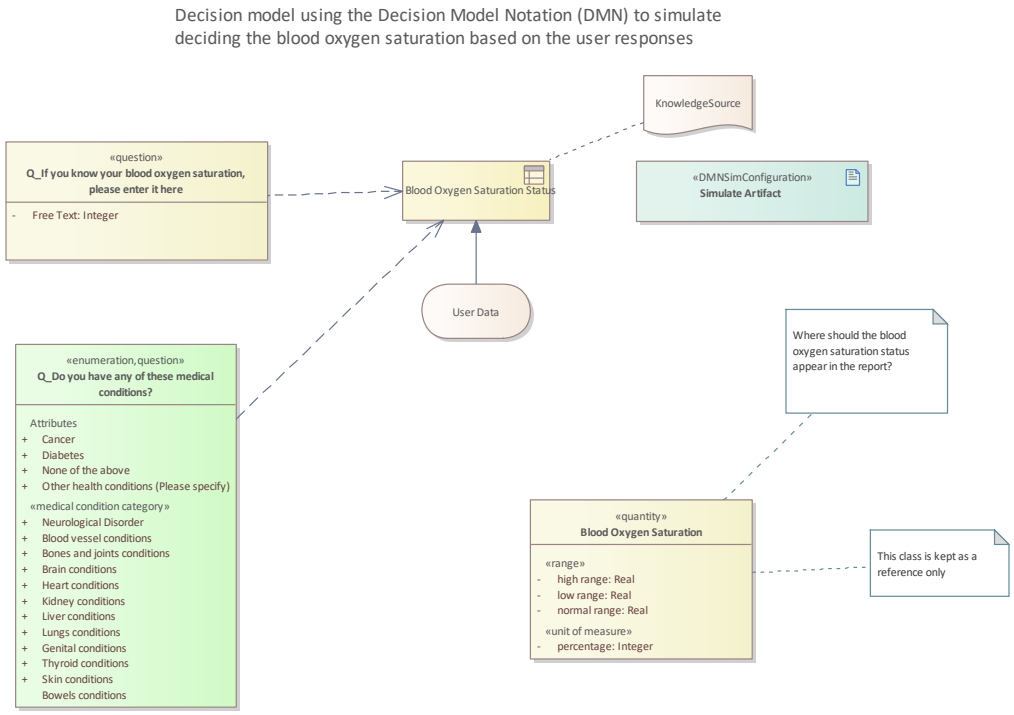


Figure 2: DecisionTable

1.1.3 DecisionTable diagram

Decision Requirements diagram in package 'Diabetes Risk Result'

DecisionTable
Version 1.0
Author: Moe Huss

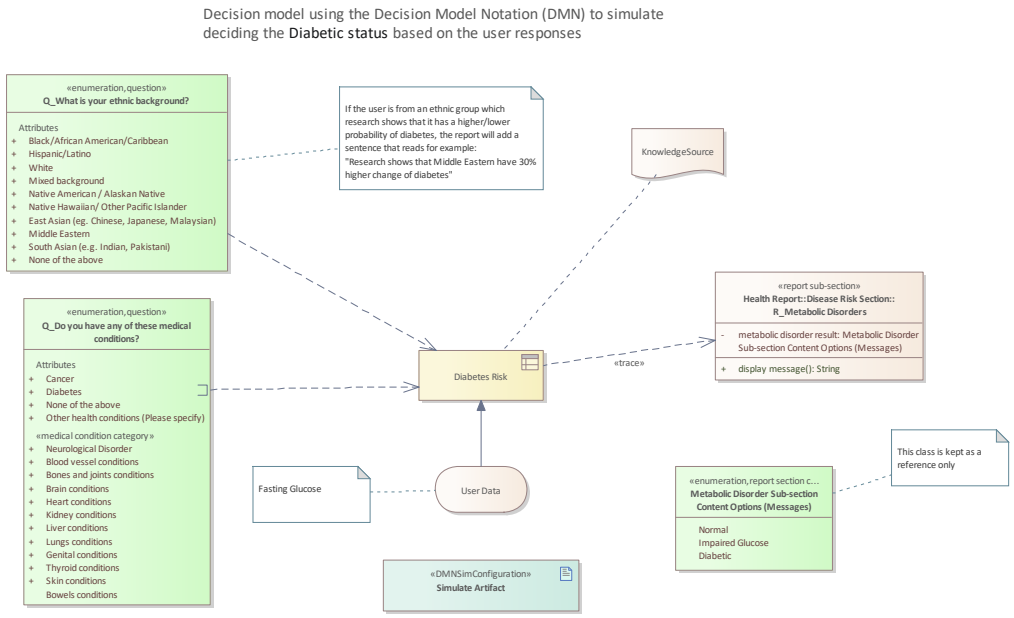


Figure 3: DecisionTable

Diagram Report

h. Quantities and Units of Measure

Version 1.0 • Final

Date Generated: 2/21/2022

Author: Moe Huss

Table of Contents

1.1.1 Basic Metabolic Panel diagram

Class diagram in package 'h. Quantities and Units of Measure'

Basic Metabolic Panel
Version 1.0
Author: Moe Huss

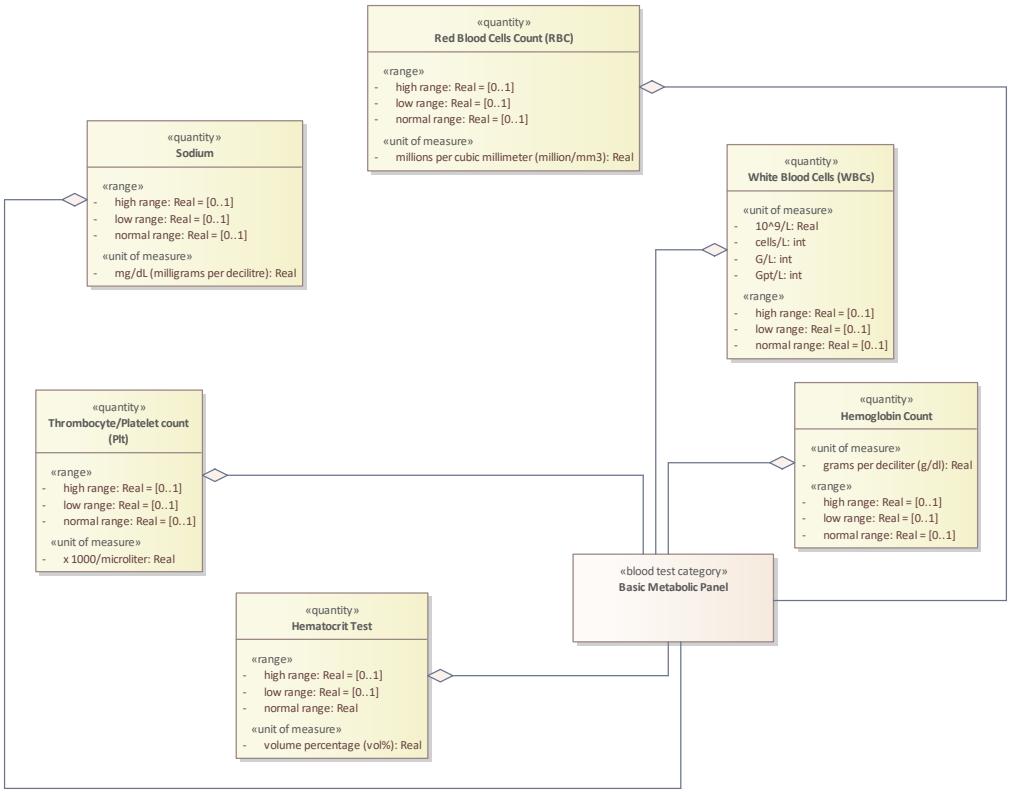


Figure 1: Basic Metabolic Panel

1.1.2 Complete Metabolic Panel diagram

Class diagram in package 'h. Quantities and Units of Measure'

Complete Metabolic Panel
Version 1.0
Author: Moe Huss

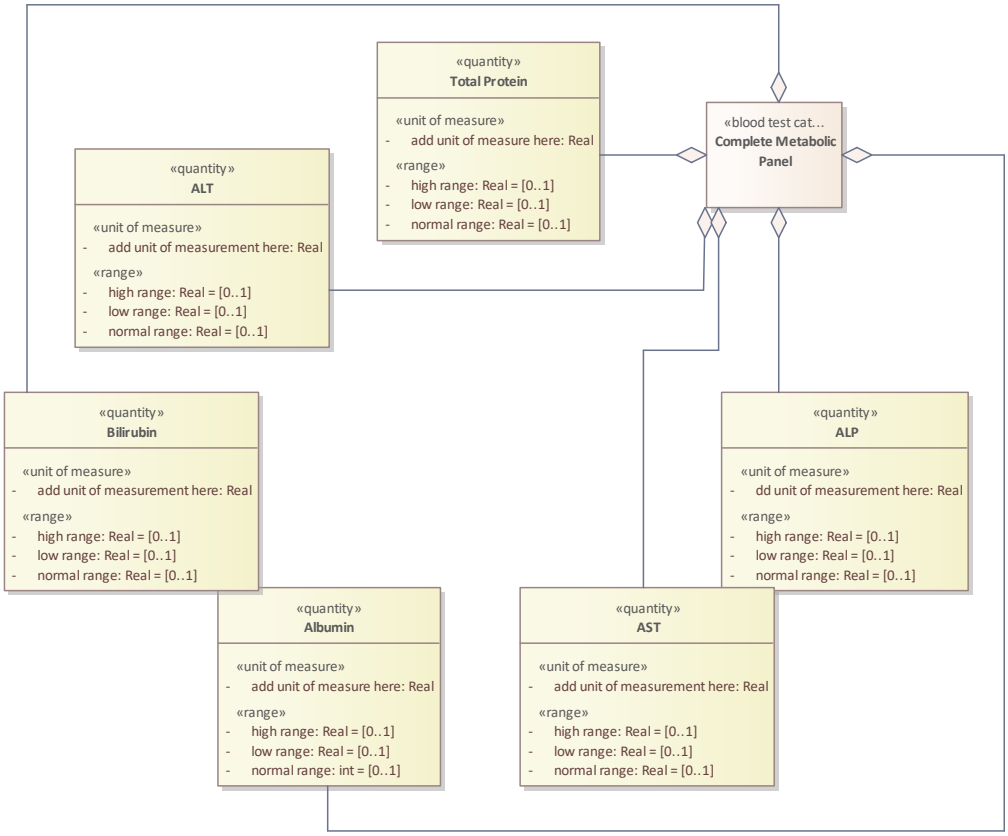


Figure 2: Complete Metabolic Panel

1.1.3 Lipid Profile diagram

Class diagram in package 'h. Quantities and Units of Measure'

Lipid Profile
Version 1.0
Author: Moe Huss

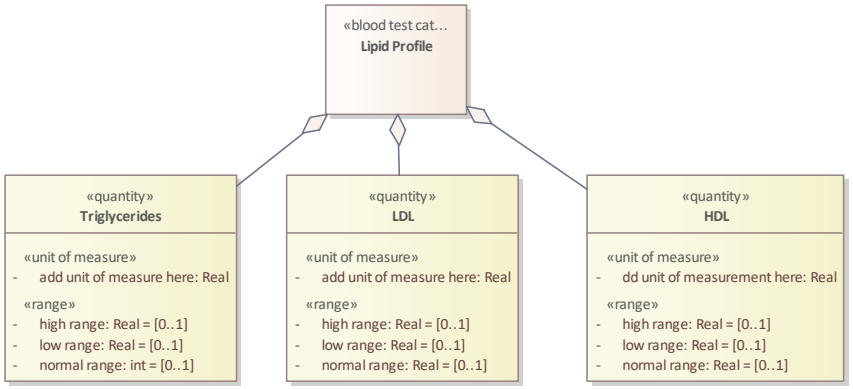


Figure 3: Lipid Profile

1.1.4 Nutrient tests for levels of vital nutrients diagram

Class diagram in package 'h. Quantities and Units of Measure'

Nutrient tests for levels of vital nutrients
Version 1.0
Author: Moe Huss

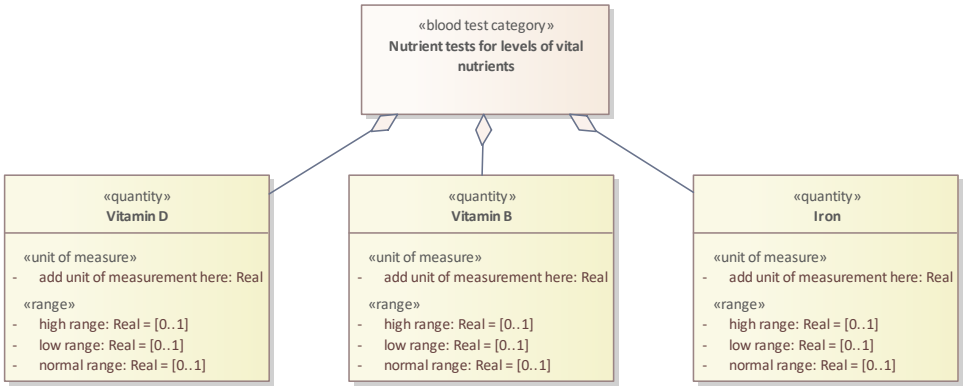


Figure 4: Nutrient tests for levels of vital nutrients

1.1.5 Thyroid Function Tests diagram

Class diagram in package 'h. Quantities and Units of Measure'

Thyroid Function Tests
Version 1.0
Author: Moe Huss

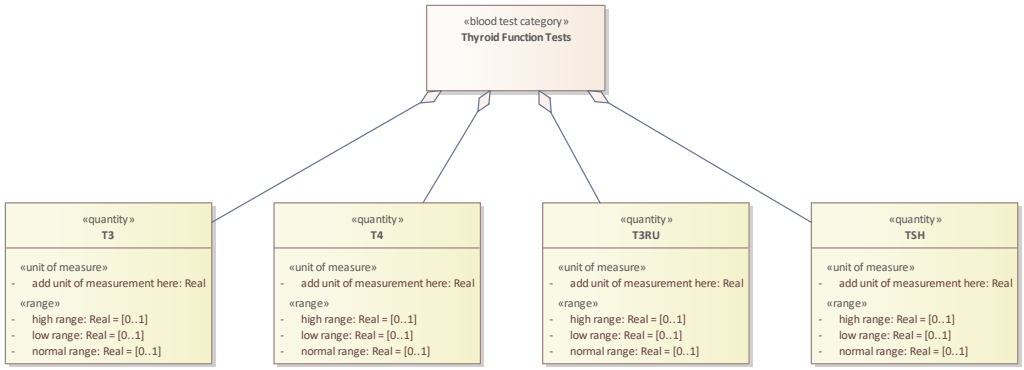


Figure 5: Thyroid Function Tests

1.1.6 Body Measurement diagram

Class diagram in package 'h. Quantities and Units of Measure'

Body Measurement
Version 1.0
Author: Moe Huss

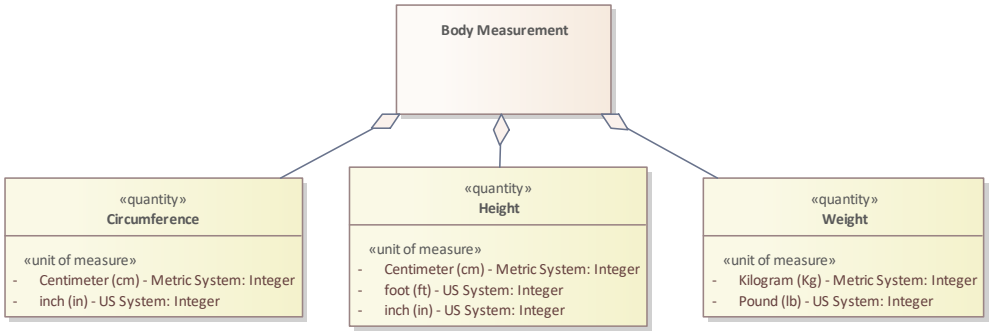


Figure 6: Body Measurement

Appendix B

Sample Sprint Log

This appendix includes a sample sprint log used to develop the health tech app. For intellectual property reasons, not all sprint logs are included.

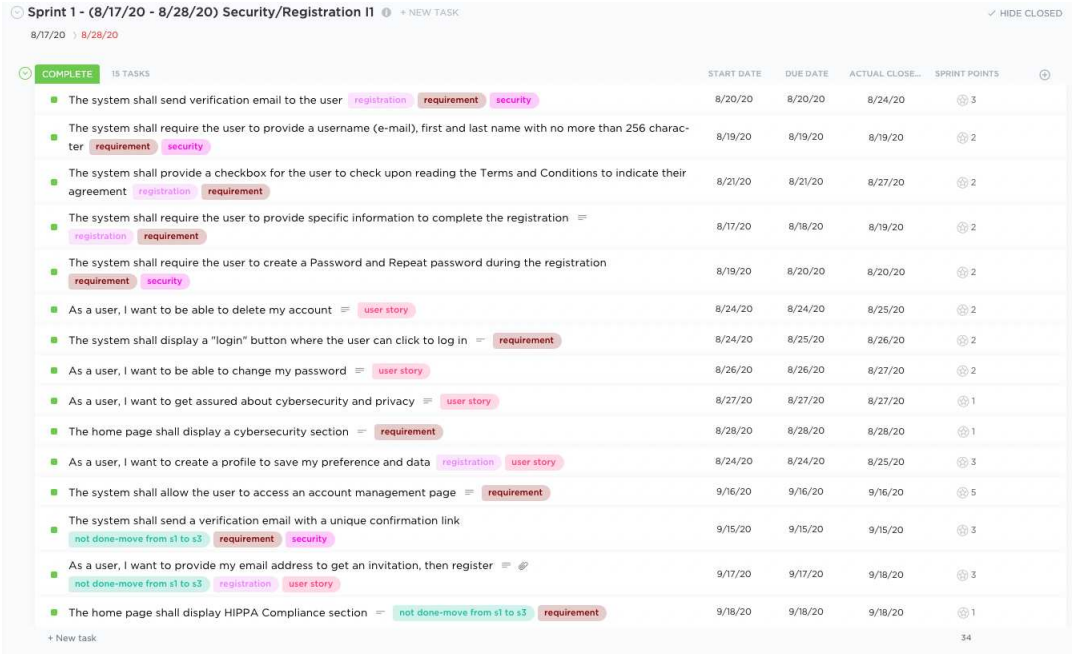


Figure B.1: Sprint 1 log — Scrum.

Appendix C

Sprint Burnup Charts

This appendix includes the sprint burnup charts used to monitor the progress of the software development of the health tech app.

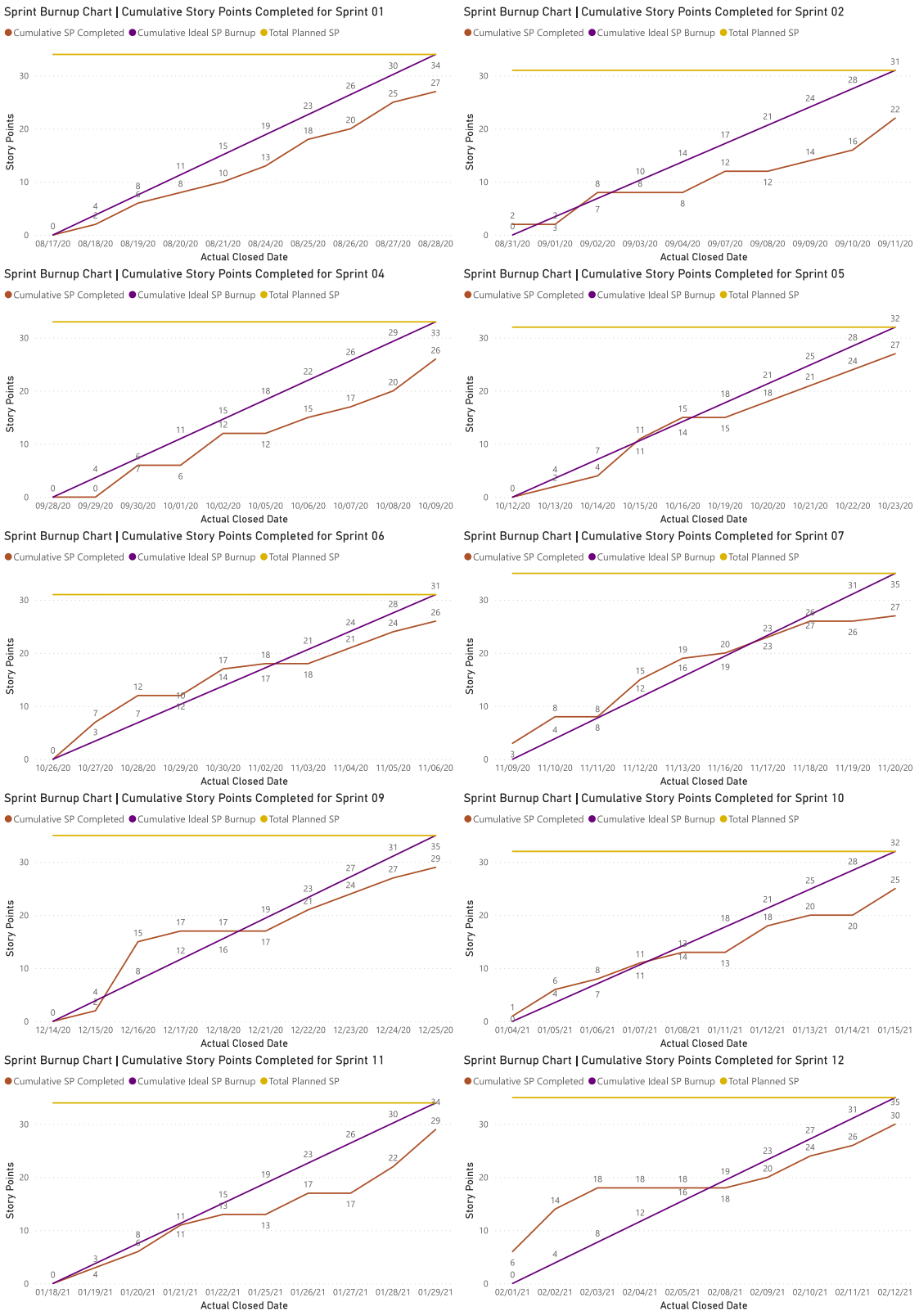


Figure C.1: Scrum burnup charts.

Figure C.2: sMBSAP burnup charts.

Appendix D

Additional Data and Statistical Analysis Figures

This appendix includes additional tables and charts with raw data and statistical analysis of the experiment.

Table D.1: Monitoring CR for Scrum-driven sprints.

Sprint #	Start Date	End Date	Planned SP	Completed SP
Sprint 01	8/17/2020	8/28/2020	34	27
Sprint 02	8/31/2020	9/11/2020	31	22
Sprint 04	9/28/2020	10/9/2020	33	26
Sprint 05	10/12/2020	10/23/2020	32	27
Sprint 06	10/26/2020	11/6/2020	31	26
Sprint 07	11/9/2020	11/20/2020	35	27
Sprint 09	12/14/2020	12/25/2020	35	29
Sprint 10	01/04/2021	1/15/2021	32	25
Sprint 11	01/18/2021	1/29/2021	34	29
Sprint 12	02/01/2021	2/12/2021	35	30

Table D.2: Monitoring CR for sMBSAP-driven sprints.

Sprint #	Start Date	End Date	Planned SP	Completed SP
Sprint 03	9/14/2020	9/25/2020	31	27
Sprint 08	11/30/2020	12/11/2020	36	34
Sprint 13	02/15/2021	2/26/2021	36	34
Sprint 14	03/1/2021	3/12/2021	34	32
Sprint 15	03/15/2021	3/26/2021	32	30
Sprint 16	3/29/2021	4/9/2021	36	33
Sprint 17	04/12/2021	4/23/2021	34	31
Sprint 18	04/26/2021	5/7/2021	34	32
Sprint 19	05/10/2021	5/21/2021	35	33
Sprint 20	05/24/2021	5/28/2021	18	18

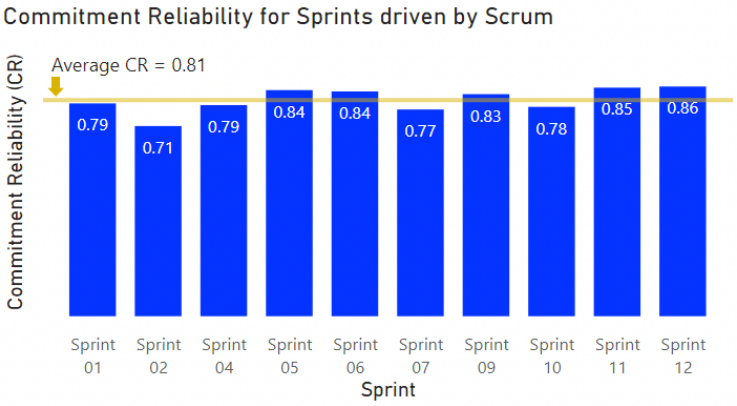


Figure D.1: Commitment Reliability for Scrum-driven sprints.

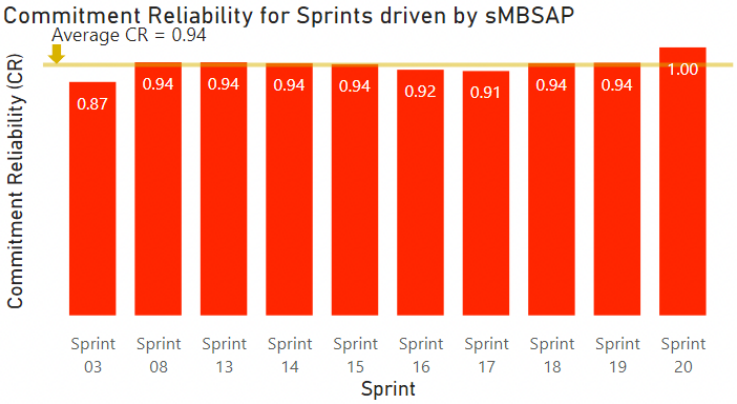


Figure D.2: Commitment Reliability for sMBSAP-driven sprints.

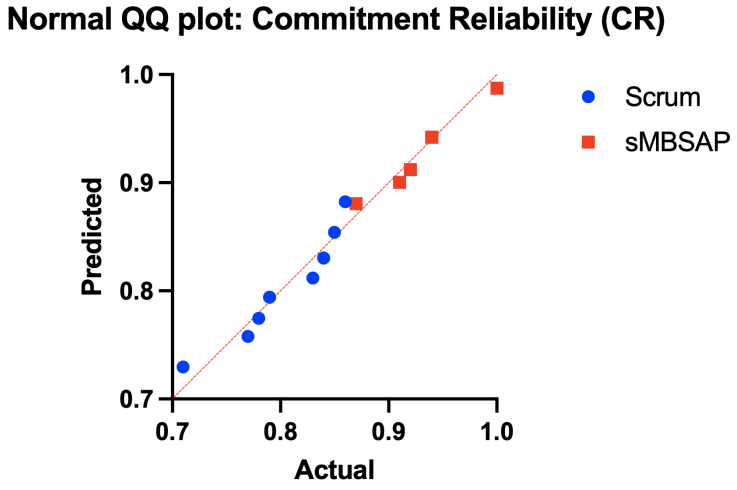


Figure D.3: Normal QQ Plot for Commitment Reliability (Normality Test).

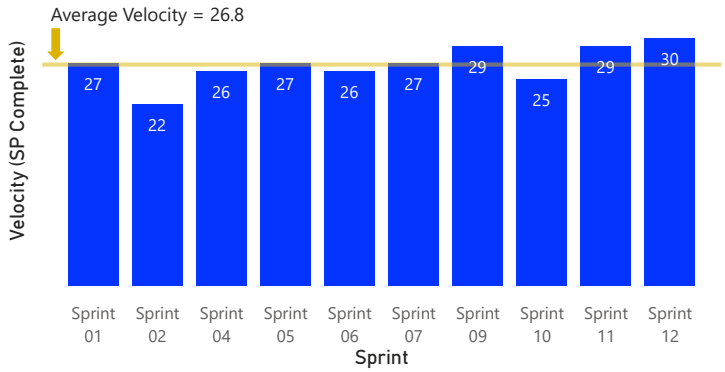


Figure D.4: Sprint Velocity for Scrum-driven sprints.

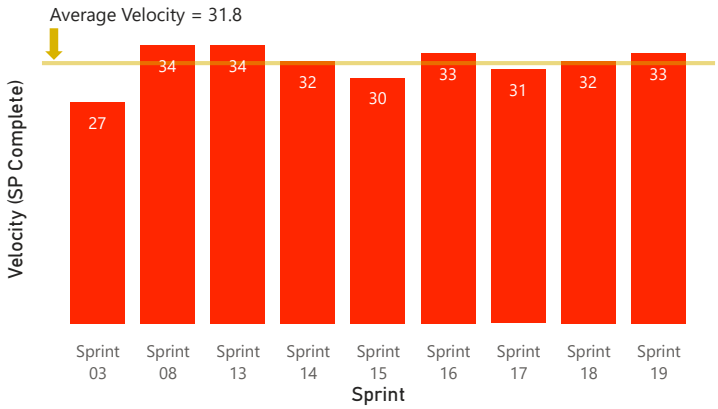


Figure D.5: Sprint Velocity for sMBSAP-driven sprints.

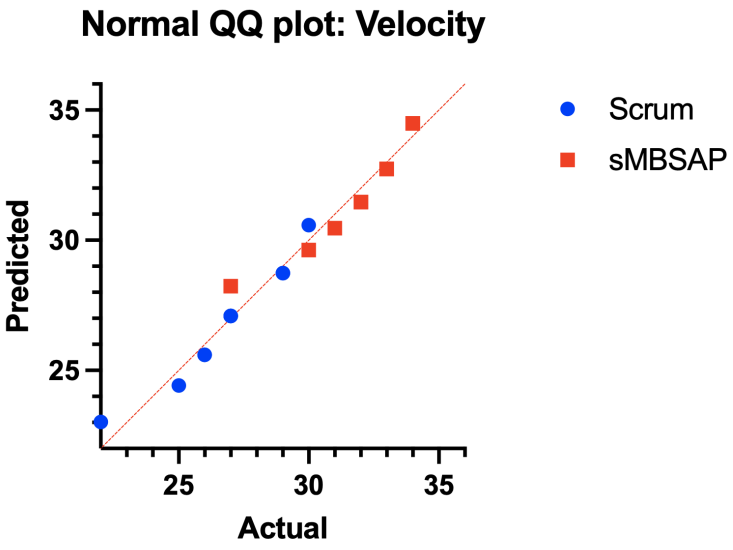


Figure D.6: Normal QQ plot for Sprint Velocity (Normality Test).

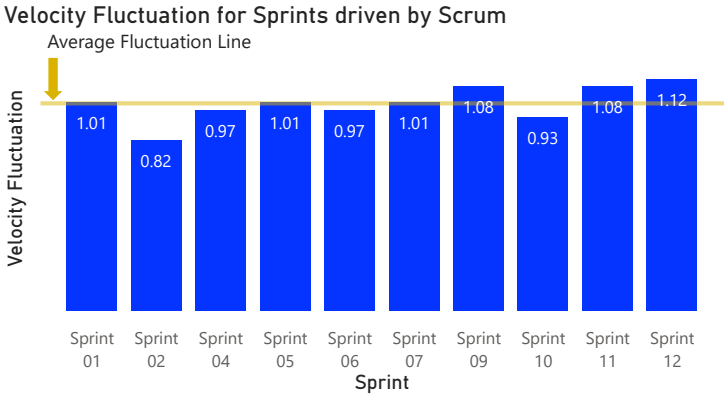


Figure D.7: Velocity Fluctuation for Scrum-driven sprints.

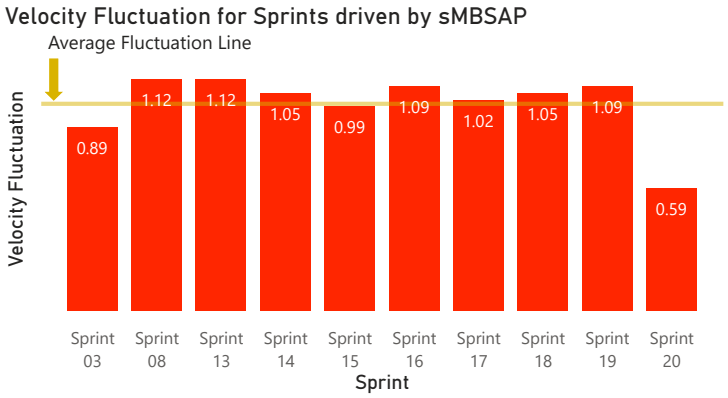


Figure D.8: Velocity Fluctuation for sMBSAP-driven sprints.

Table D.3: Monitoring Defects for Scrum-driven sprints.

Sprint #	PBI	Pre-Delivery Defects	Post-Delivery Defects
Sprint 01	16	18	3
Sprint 02	15	17	3
Sprint 04	16	13	3
Sprint 05	13	13	3
Sprint 06	42	31	6
Sprint 07	20	16	3
Sprint 09	26	28	5
Sprint 10	23	24	5
Sprint 11	27	19	4
Sprint 12	41	28	5

Table D.4: Monitoring Defects for sMBSAP-driven sprints.

Sprint #	PBI	Pre-Delivery Defects	Post-Delivery Defects
Sprint 03	17	6	1
Sprint 08	26	24	3
Sprint 13	32	10	2
Sprint 14	15	6	1
Sprint 15	16	14	3
Sprint 16	11	10	2
Sprint 17	17	13	2
Sprint 18	8	7	1
Sprint 19	18	12	2
Sprint 20	24	4	0

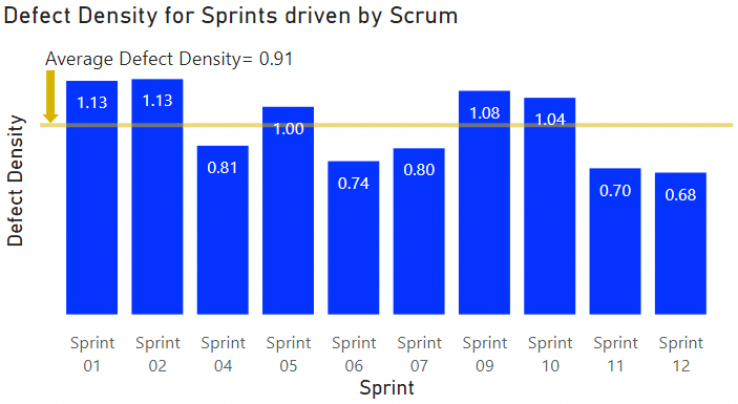


Figure D.9: Defect Density for Scrum-driven sprints.

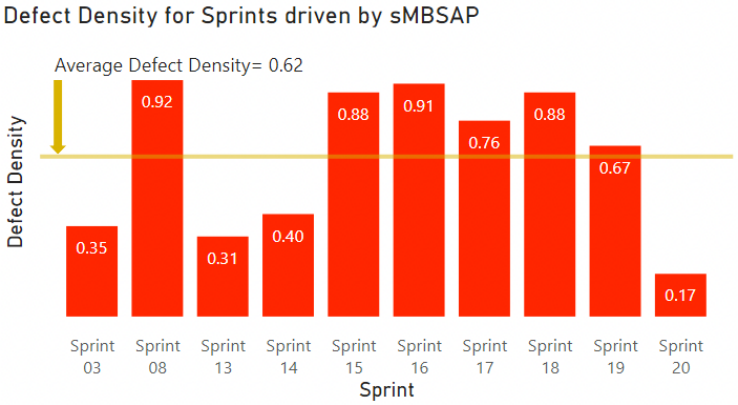


Figure D.10: Defect Density for sMBSAP-driven sprints.

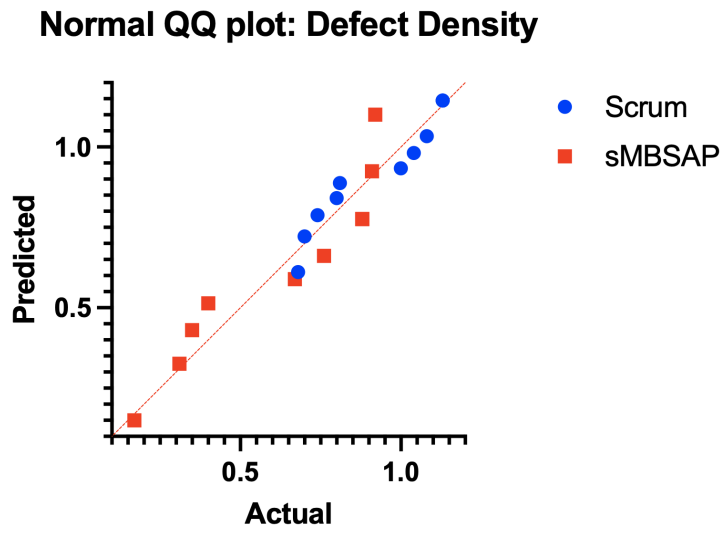


Figure D.11: Normal QQ plot for Defect Density (Normality Test).

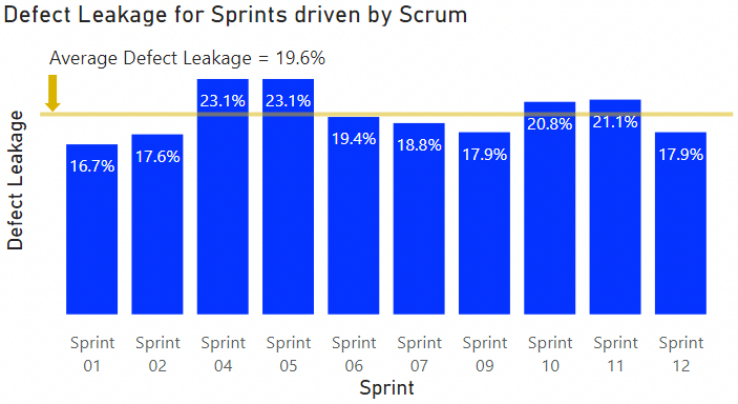


Figure D.12: Defect Leakage for Scrum-driven sprints.

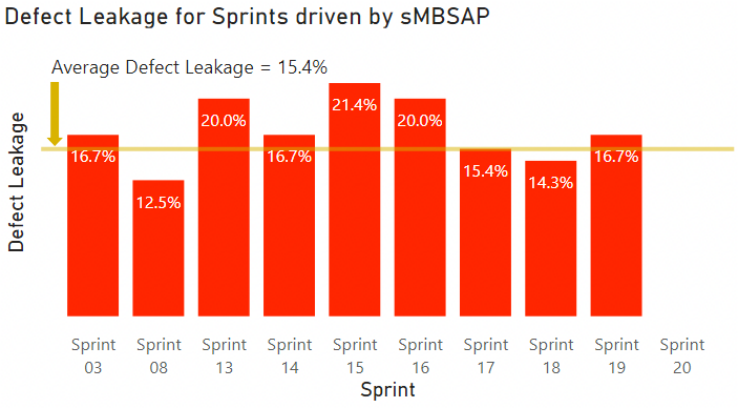


Figure D.13: Defect Leakage for sMBSAP-driven sprints.

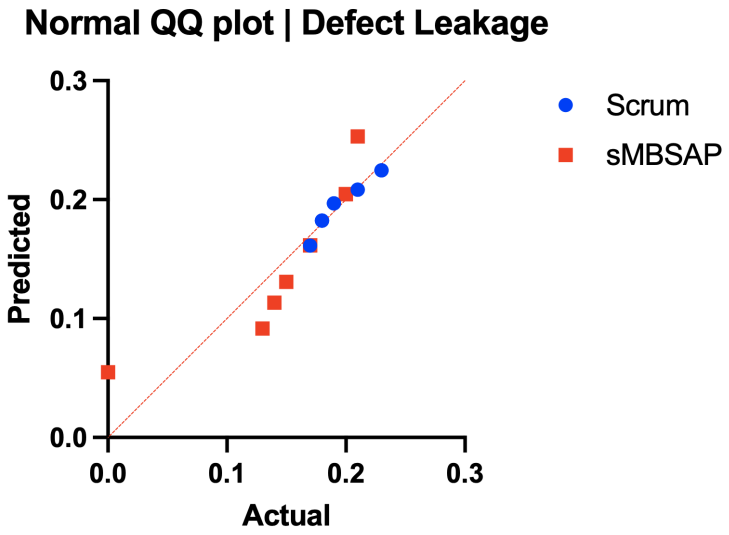


Figure D.14: Normal QQ plot for Defect Leakage (Normality Test).

Appendix E

Health Tech System Screenshots and User Personas

This appendix includes screenshots of the health tech app and the personas of the users.

E.1 Health Tech System Screenshots

Figure E.1: System demo experience guide.

Figure E.2: System demo steps.

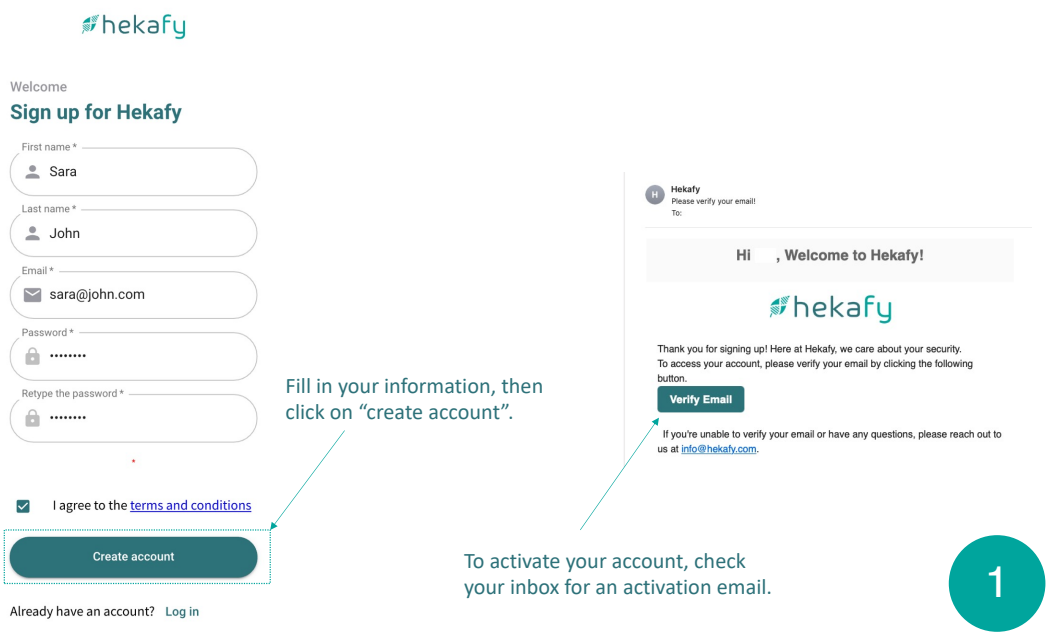


Figure E.3: Create account screen.

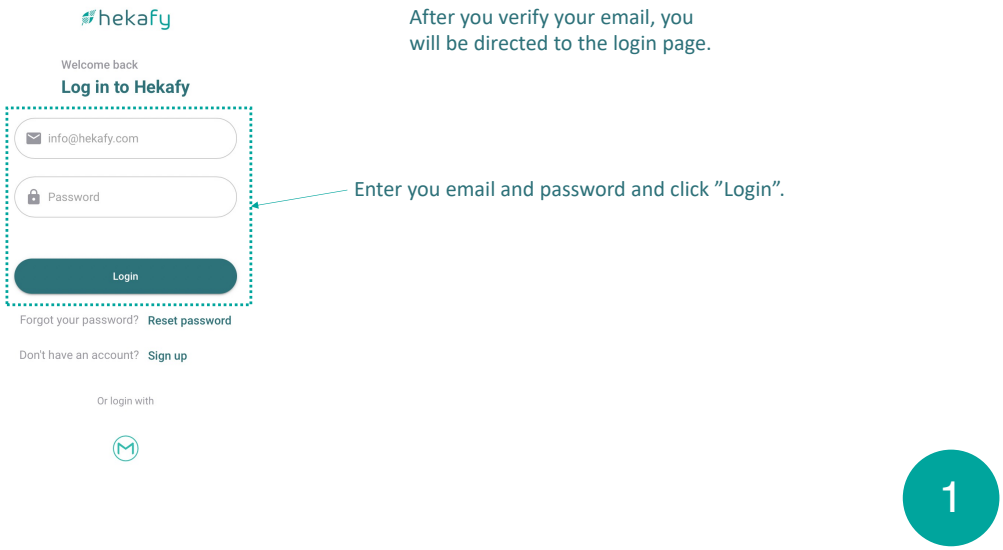


Figure E.4: Login screen.

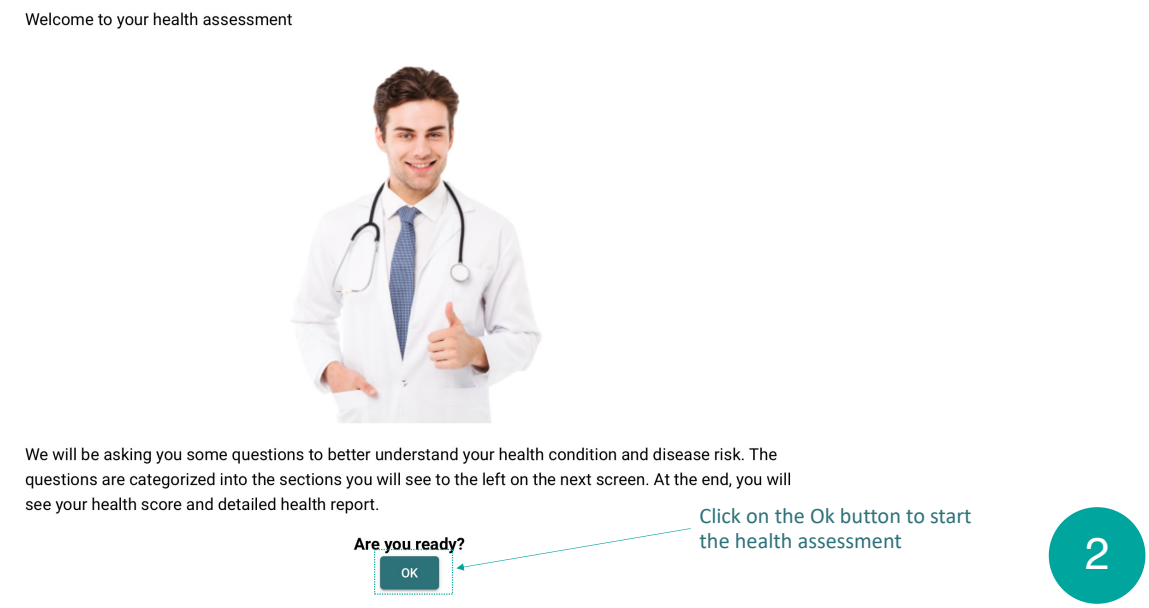


Figure E.5: Health assessment welcome screen.

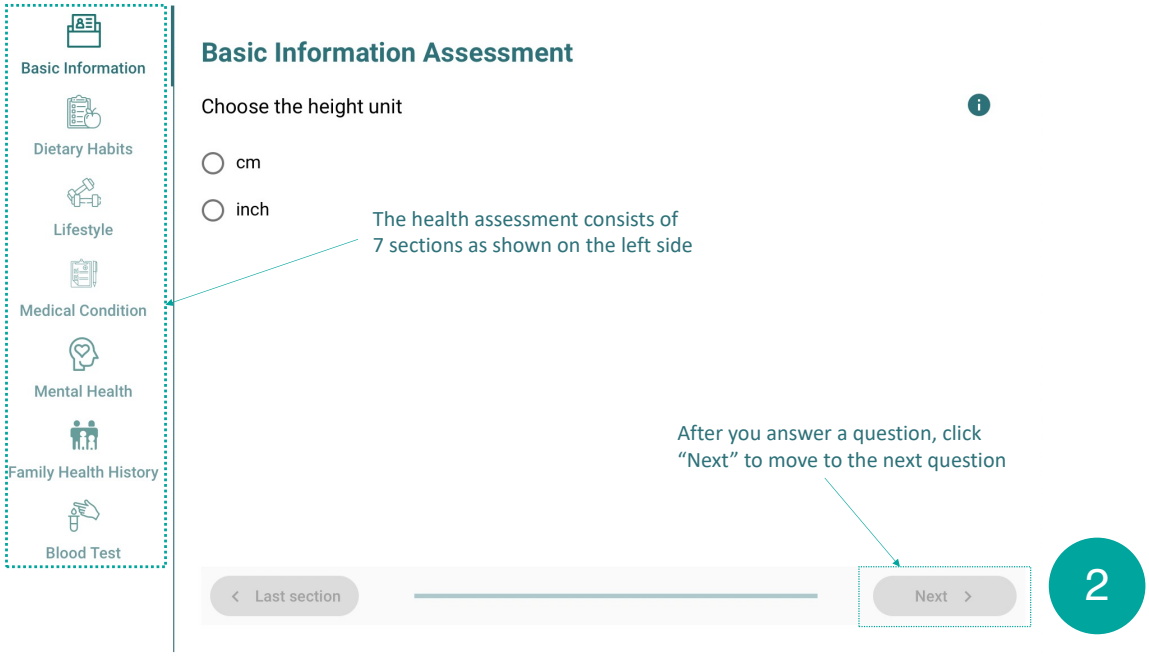


Figure E.6: Health assessment question screen.

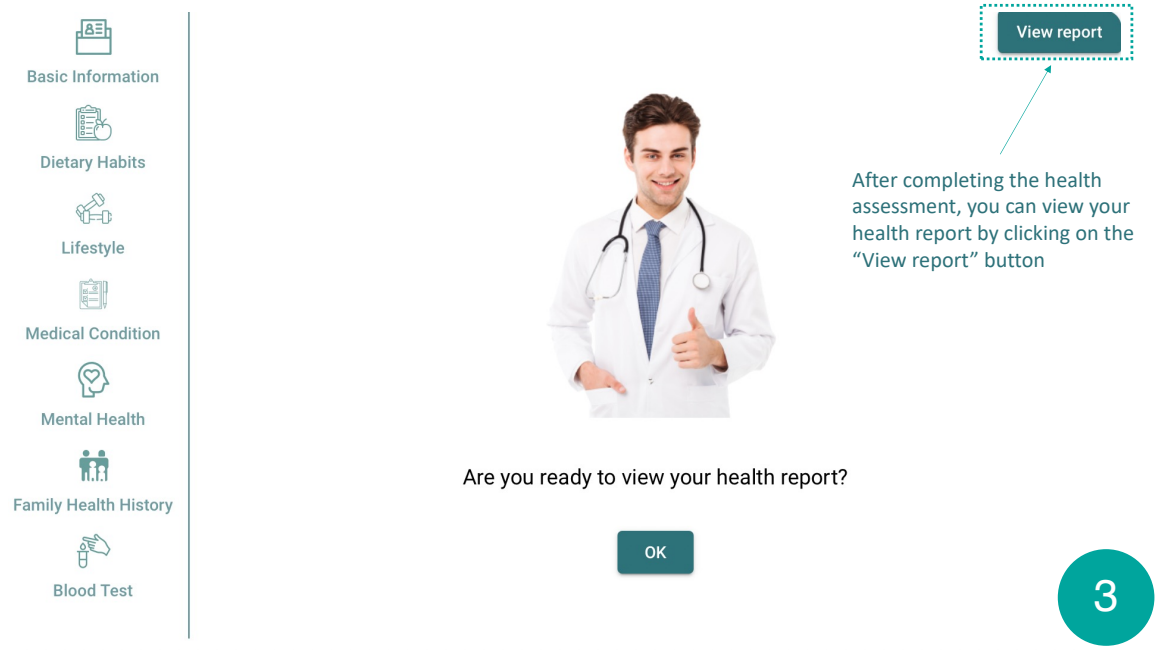


Figure E.7: End of health assessment.

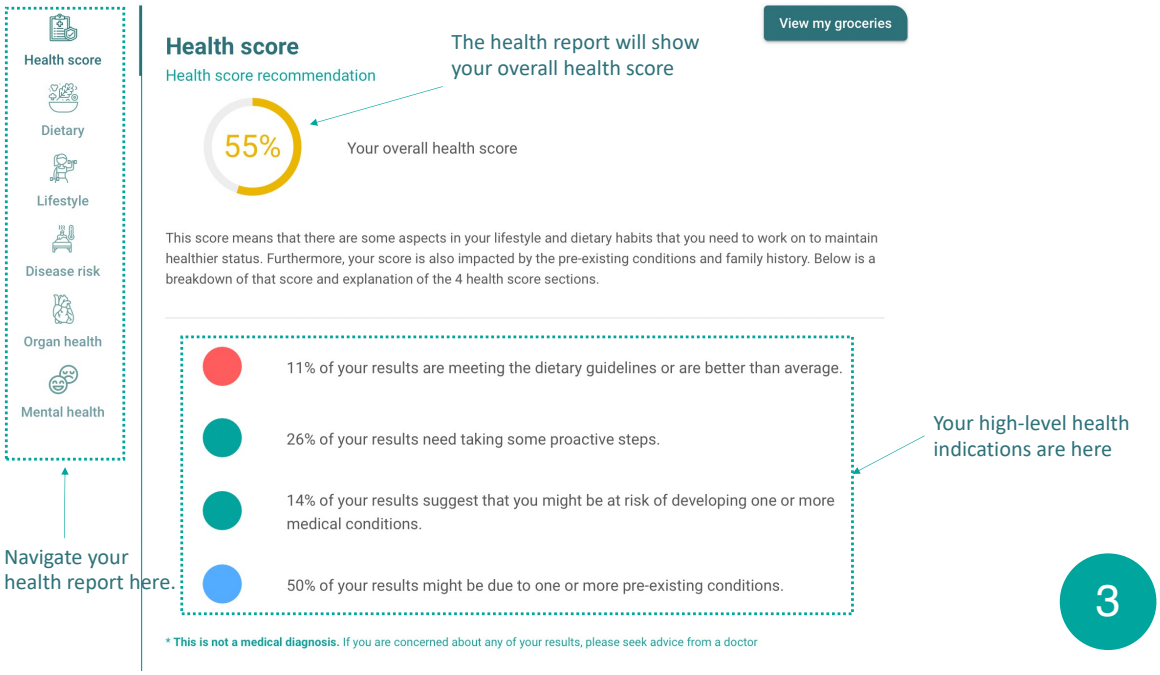


Figure E.8: Health report – health score screen.

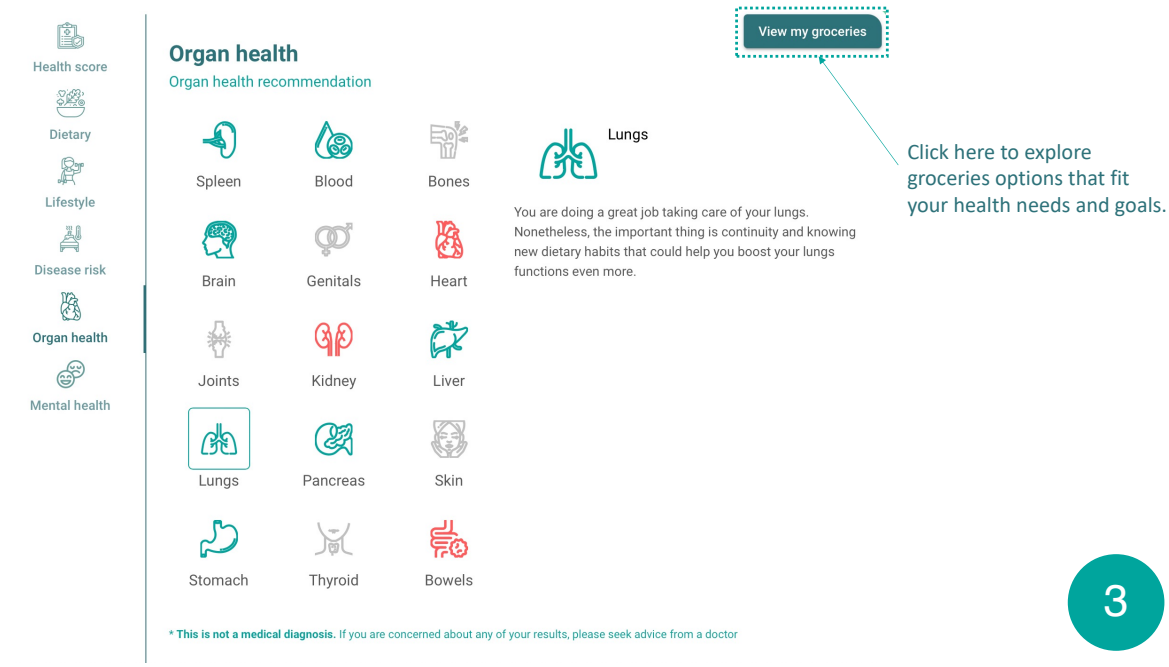


Figure E.9: Health report – organ health screen.

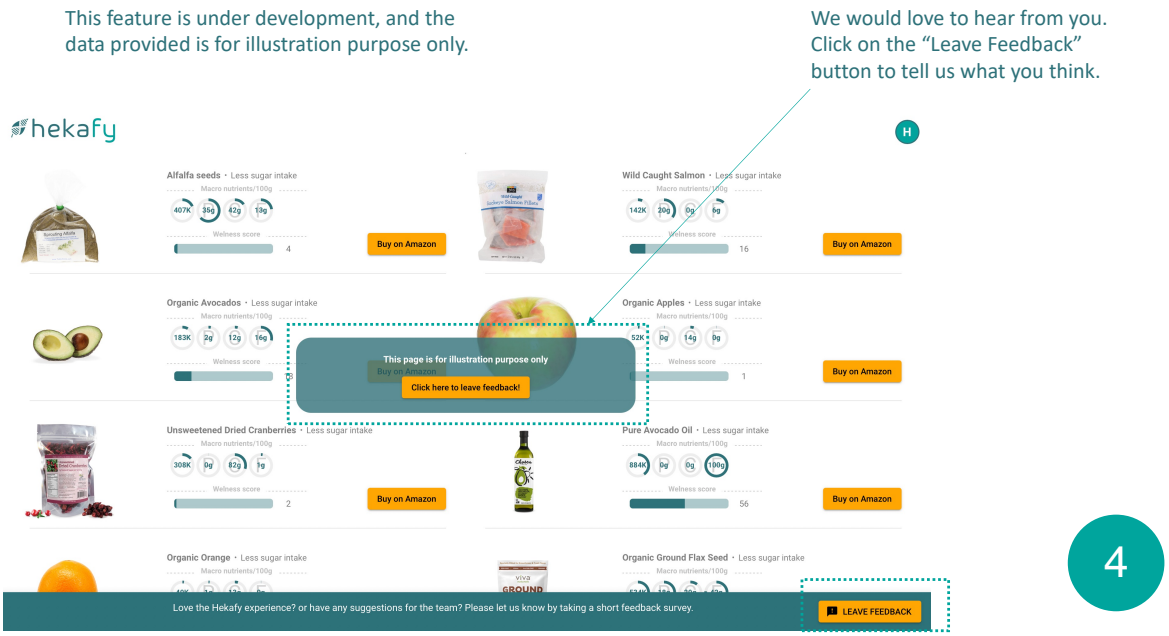


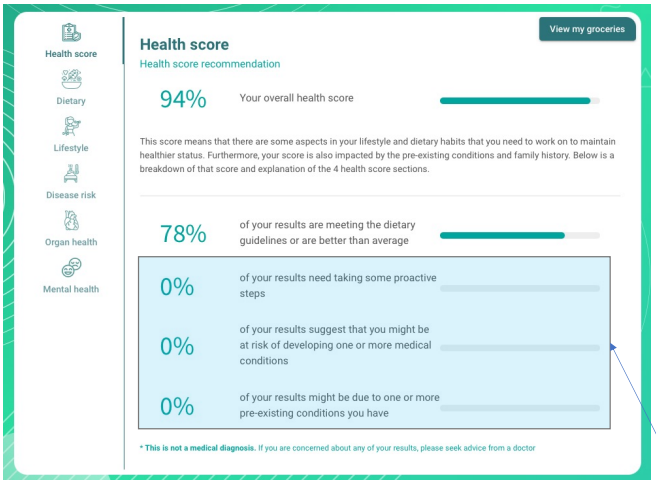
Figure E.10: Grocery recommendations screen.

hekafy

Thank You

www.hekafy.com

Figure E.11: End of system demo.



The Problem

The problem we are trying to solve is the confusion that would happen to the user to interpret the health report.

Example

A healthy user who sees three 0% without any bar progress would either think that the report has not been populated or the results are incorrect.

Solution

The following three slides suggest a potential solution

Figure E.12: The problem and solution.

E.2 User Personas

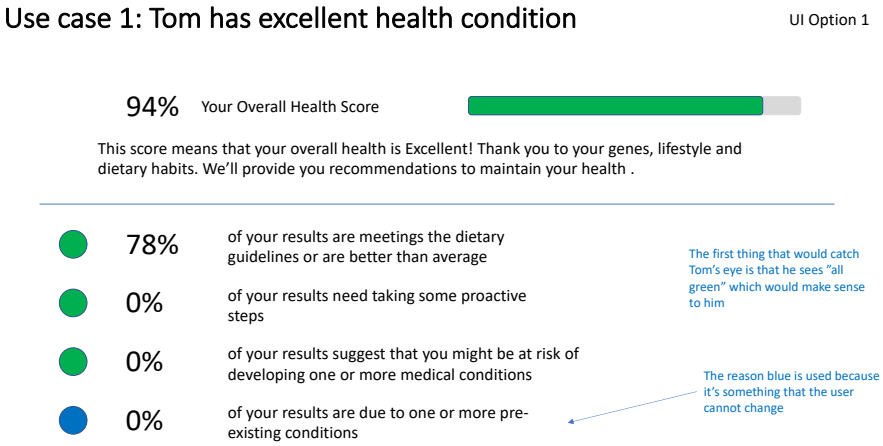


Figure E.13: Use Case 1 – Tom has excellent health condition | User Interface option 1.

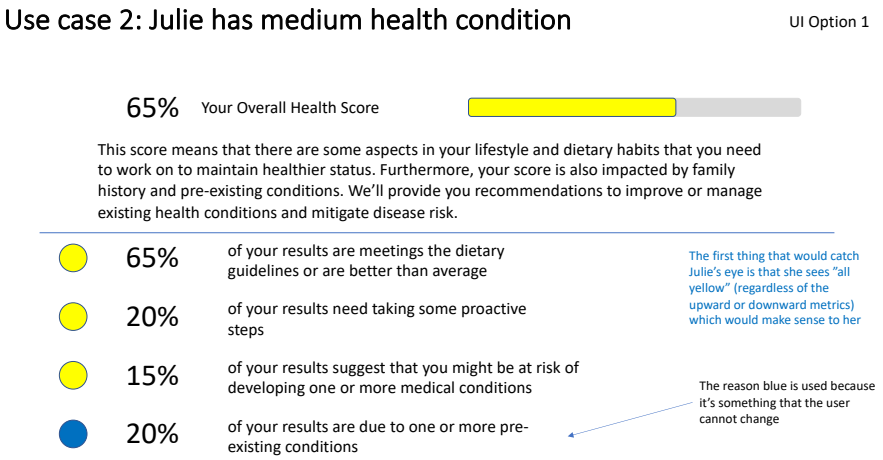


Figure E.14: Use Case 2 – Julie has medium health condition | User Interface option 1.

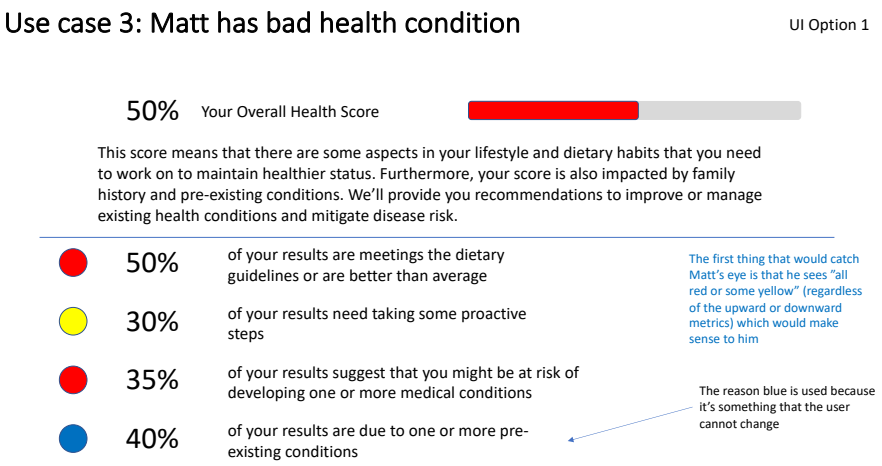


Figure E.15: Use Case 3 – Matt has bad health condition | User Interface option 1.

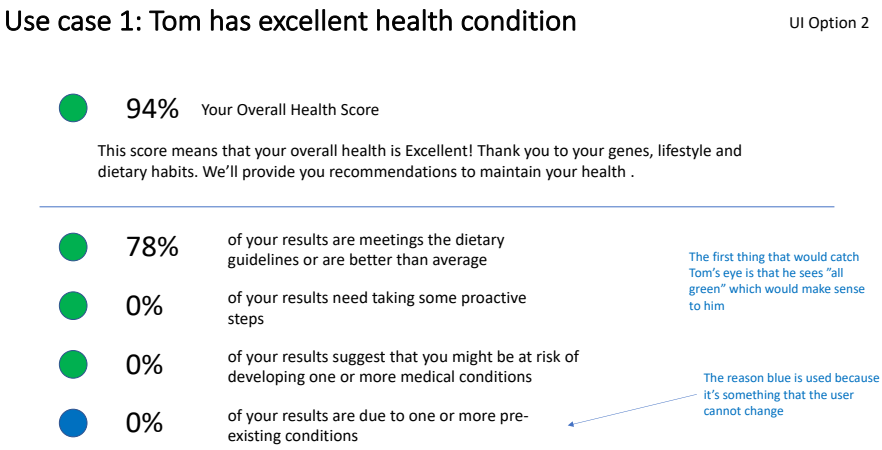


Figure E.16: Use Case 1 – Tom has excellent health condition | User Interface option 2.

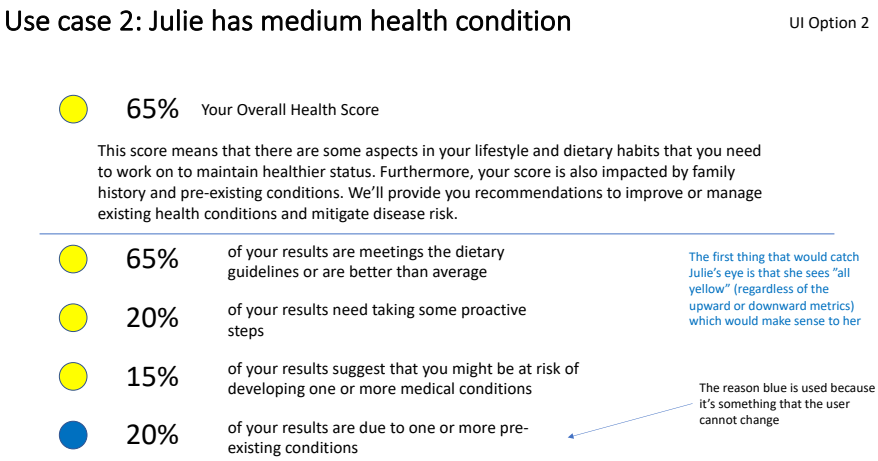


Figure E.17: Use Case 2 – Julie has medium health condition | User Interface option 2.

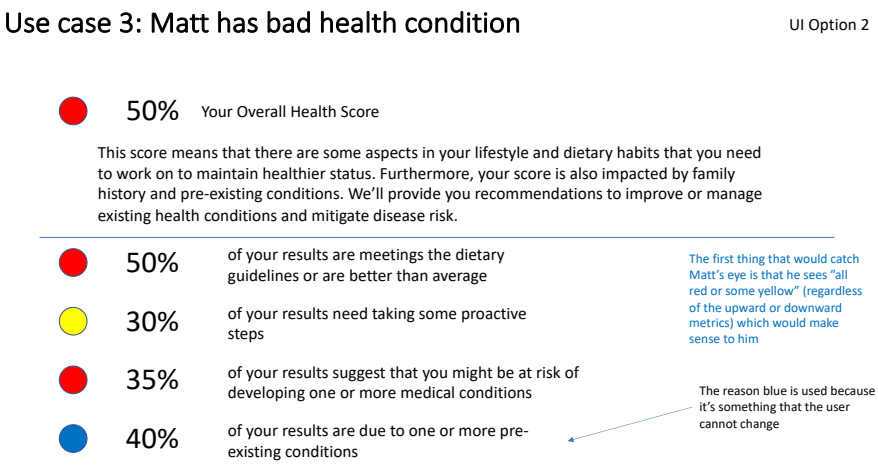


Figure E.18: Use Case 3 – Matt has Bad health condition | User Interface option 2.

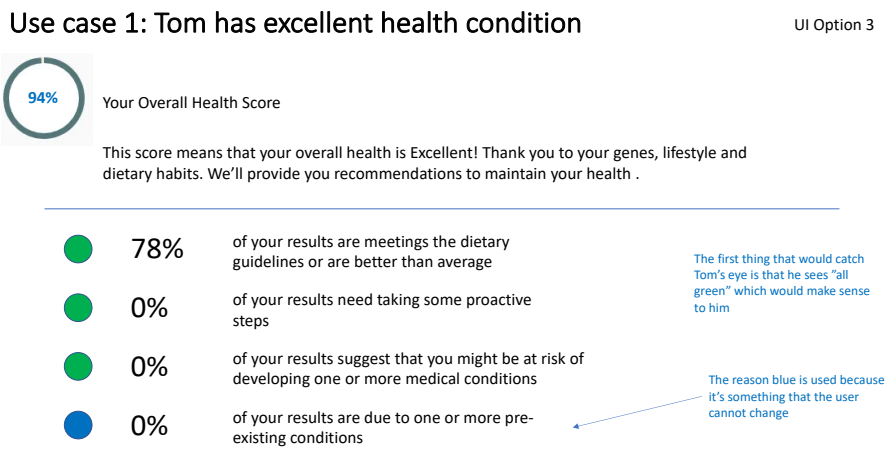


Figure E.19: Use Case 1 – Tom has excellent health condition | User Interface option 3.

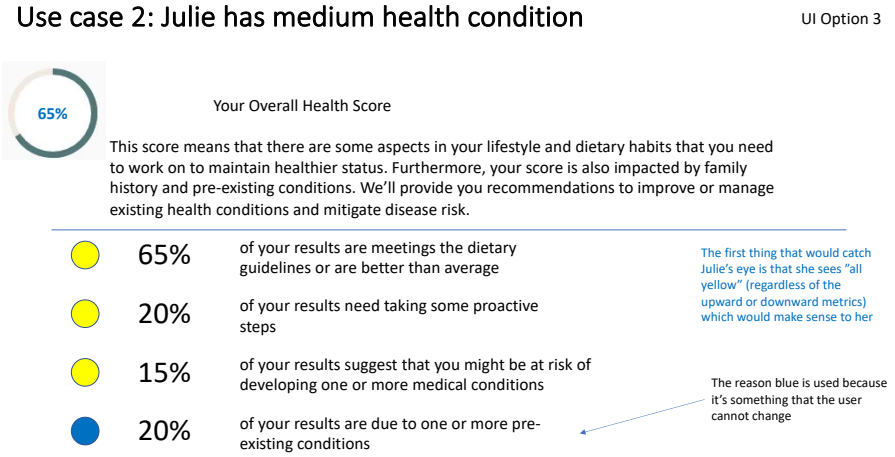


Figure E.20: Julie has medium health condition | User Interface option 3.

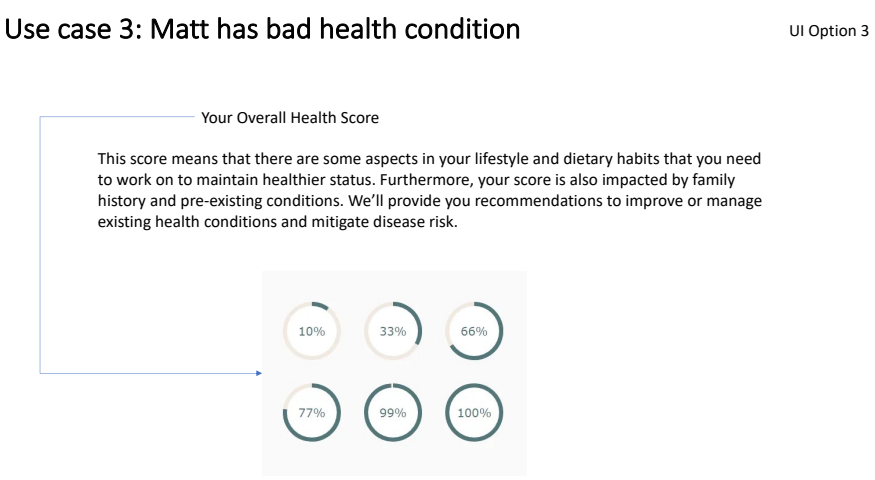


Figure E.21: Use Case 3 – Matt has bad health condition | User Interface option 3.

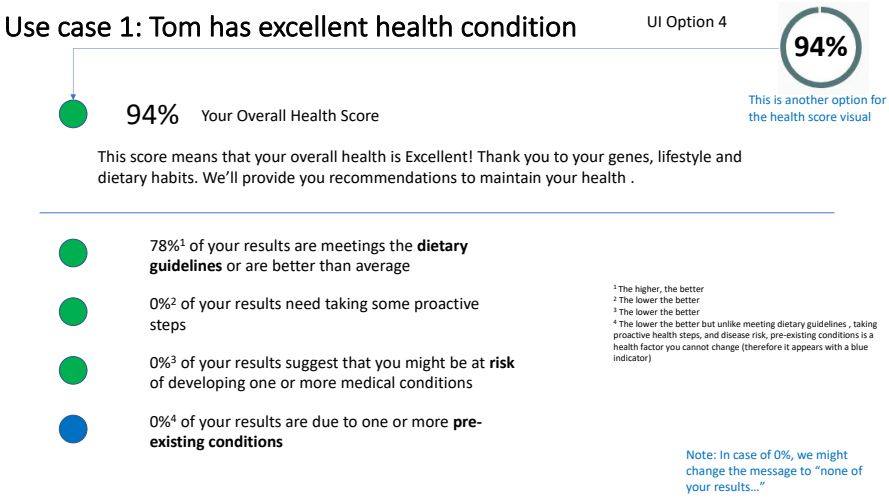


Figure E.22: Use Case 3 – Tom has excellent health condition | User Interface option 4.

Acronyms

AC Actual Cost *on page(s):* 10

AD Activity Diagrams *on page(s):* ix, 62, 63

AHA American Heart Association *on page(s):* 16

AMDR Acceptable Macronutrient Distribution Range *on page(s):* 15

ASD Adaptive Software Development *on page(s):* 32

BE Back-end *on page(s):* 19

CDC The Center for Disease Control and Prevention *on page(s):* 15

CDM Conceptual Data Model *on page(s):* ix, 41–43, 63

CLOC Count of Lines of Code *on page(s):* 11, 14, 60, 67, 69, 73, 83, 86–88, 91, 93, 94, 97

CR Commitment Reliability *on page(s):* iii, viii, 9, 13, 66, 83, 84, 91, 97, 268

DBMS Database Management System *on page(s):* 43, 44

DBSE Document-Based Systems Engineering *on page(s):* 5, 27, 29, 30, 34

DD Defect Density *on page(s):* iii, viii, ix, 12, 14, 83, 87–89, 91, 93–95, 97, 272, 273

DL Defect Leakage *on page(s):* iii, viii, ix, 12, 14, 83, 88, 89, 91, 93, 273, 274

DoD Department of Defense *on page(s):* 10

EV Earned Value *on page(s):* 10

EVM Earned Value Management *on page(s):* 10

FE Front-end *on page(s):* 19

GDP Gross Domestic Product *on page(s):* 15

HA Health Assessment *on page(s):* 21

HIPPA Health Insurance Portability and Accountability Act *on page(s):* 20

HR Health Reporting *on page(s):* 21

IEC International Electrotechnical Commission *on page(s):* 2

IEEE Institute of Electrical and Electronics Engineers *on page(s):* 2

INCOSE International Council on Systems Engineering *on page(s):* 30

ISO International Organization for Standardization *on page(s):* 2

IT Information Technology *on page(s):* 4

KLOC Thousands (Kilo) of Lines of Code *on page(s):* 12, 14, 59, 68, 88, 91, 93, 95, 97

LDM Logical Data Model *on page(s):* ix, 43, 44, 63

LV/FV Logical/Functional Viewpoint *on page(s):* 31, 43, 52, 92, 95

MAGIC Munich Agile MBSE Concept *on page(s):* 33

MBSAP Model-Based System Architecture Process *on page(s):* ii, ix, 2, 8, 23, 26, 29, 31, 32, 35, 46, 52, 53, 63, 66, 76–78, 91, 92, 95

MBSE Model-Based Systems Engineering *on page(s):* ii, iii, ix, 1–3, 6–8, 12, 13, 21–24, 27–31, 33, 34, 37, 38, 45, 48, 52, 53, 55, 66, 76, 78–80, 82, 83, 90, 92, 95–98

ML Machine Learning *on page(s):* 20

NCCs Nutrient Content Claims *on page(s):* 17

NFP Nutrition Facts panel *on page(s):* 16

OV Operational Viewpoint *on page(s):* 41, 43, 52, 82, 92, 95

PBI Product Backlog Item *on page(s):* 12, 27, 45, 68

PBIs Product Backlog Items *on page(s):* 38, 39, 41, 45, 46, 50, 59, 64, 66–69, 78, 87, 93, 94, 97

PDM Physical Data Model *on page(s):* ix, 43, 44, 47, 63

PV Planned Value *on page(s):* 10

PV Physical Viewpoint *on page(s):* 31, 43, 52, 63, 92, 95

ROI Return-on-investment *on page(s):* 12

SBIs Sprint Backlog Items *on page(s):* 14

SD Sequence Diagrams *on page(s):* 63

SDLC Software Development Lifecycle *on page(s):* 5, 7, 34

SEP Scientific Evidence Platform *on page(s):* 21

sMBSAP Scrum Model Based System Architecture Process *on page(s):* ii, iii, ix, 2, 3, 12–14, 20, 21, 23, 35–38, 45, 46, 48, 51–53, 57, 58, 62, 63, 66–68, 74, 76–99

SP Story Points *on page(s):* 11–14, 66, 68, 69, 90

SV Sprint Velocity *on page(s):* iii, viii, ix, 11, 13, 14, 60, 84, 85, 91, 270

U.S. United States *on page(s):* 8, 10

UC Use Case *on page(s):* 63

UK United Kingdom *on page(s):* 4

UML Unified Modeling Language *on page(s):* ix, 1, 29, 42–44, 48, 76, 80, 82

UX User Experience *on page(s):* 8, 20

VF Velocity Fluctuation *on page(s):* viii, ix, 11, 14, 60, 67, 83, 86, 91, 93–95, 97, 271

XP Extreme Programming *on page(s):* 32