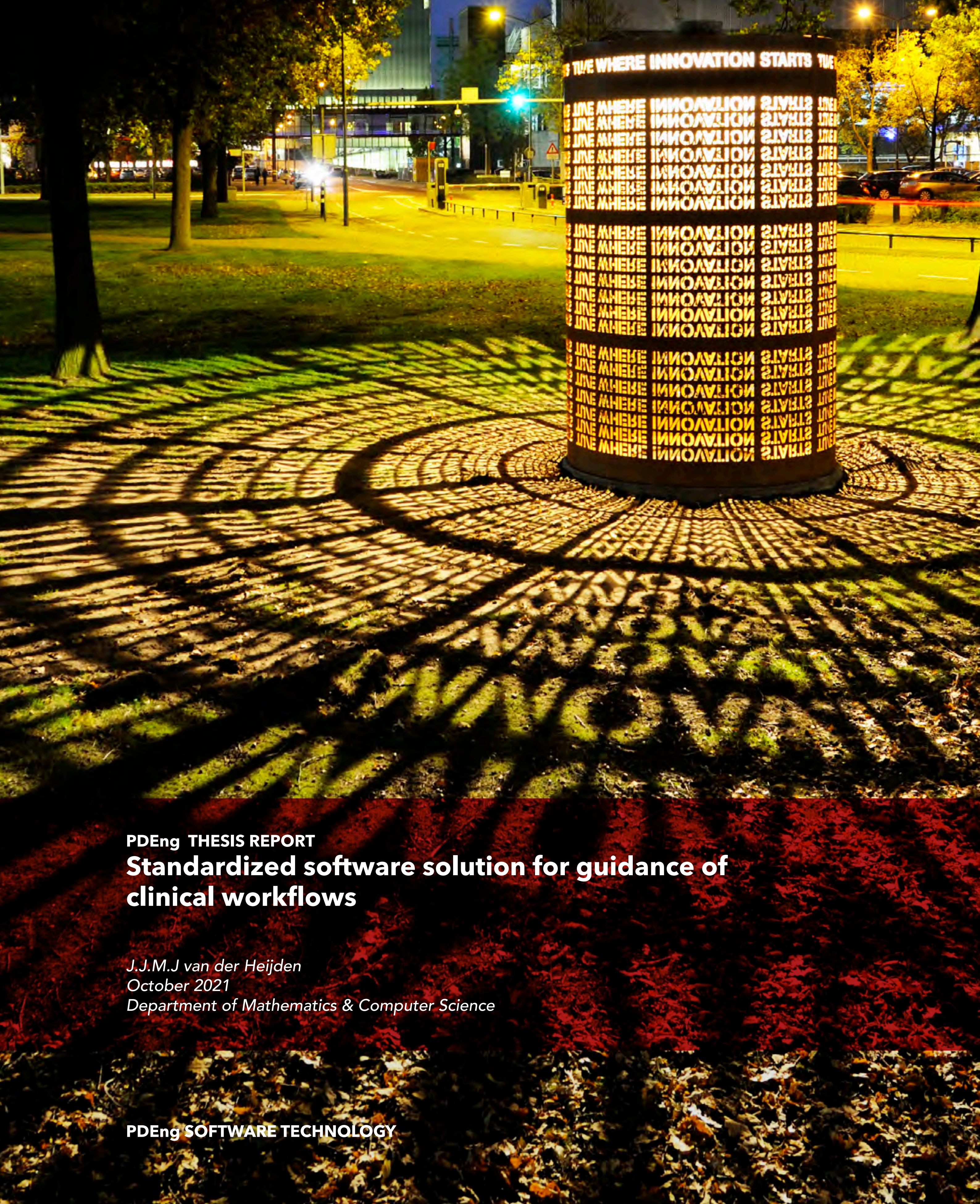


PDEng THESIS REPORT

Standardized software solution for guidance of clinical workflows

J.J.M.J van der Heijden
October 2021
Department of Mathematics & Computer Science

PDEng SOFTWARE TECHNOLOGY

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Standardized software solution for guidance of clinical workflows

Juan J.M.J. van der Heijden

October 2021

Eindhoven University of Technology
Stan Ackermans Institute – Software Technology

PDEng Report: 2021/080

Confidentiality Status:
Public

Partners	Philips Research	Eindhoven University of Technology
Steering Group	Ir. Bas Bergevoet, PDEng Prof.dr. Michel Chaudron Ir. Patrick Bonné Dr. Yanja Dajsuren, PDEng	
Date	October 2021	

Composition of the Thesis Evaluation Committee:

Chair:	Prof.dr. Michel Chaudron
Members:	Ir. Bas Bergevoet, PDEng
	Ir. Patrick Bonné
	Ir. Erik Moll
	Dr. Renata Medeiros de Carvalho

Date	October, 2021
Contact address	Eindhoven University of Technology Department of Mathematics and Computer Science Software Technology MF 5.080 A P.O. Box 513 NL-5600 MB Eindhoven, The Netherlands +31 402744334
Published by	Eindhoven University of Technology
PDEng Report	2021/080

Abstract

During a patient's stay in the hospital, patients traverse through multiple clinical pathways, often having to comply with one or more defined clinical workflows. Proper adherence to these workflows is often crucial for having the best clinical outcome for the patient. Philips envisions a standardized software solution for guidance of clinical workflows, called the HealthSuite Workflow Capability. Based on previous exploration, Philips has proposed a reference architecture for the HealthSuite Workflow Capability. In this work, based on the reference architecture, a detailed design and prototype has been presented. The design and prototype are based on standard definitions used in Healthcare, such as BPMN, DMN, and FHIR. In this work we present a mapping from the BPMN to FHIR standard, making interoperability with other systems possible. The modular design and prototype presented can work with off-the-shelf Workflow Engines, preventing a vendor lock-in. Finally, we recommend to continue expanding the presented design and prototype with the CMMN standard, and we recommend to explore the Clinical Quality Framework as possible improvements to some of the design elements.

Keywords	**protocol guidance, clinical workflows, healthcare, process modeling**
Preferred reference	Standardized software solution for guidance of clinical workflows. Eindhoven University of Technology, PDEng Report 2021/080, October 2021.
Partnership	This project was supported by Eindhoven University of Technology and Philips Research

Foreword

Philips is a major player in the healthcare market with a clear strategy to deal with today's trends and challenges in this market. An aspect of this strategy is to transform from a healthcare systems provider to a healthcare solution provider. E.g., instead of offering just a diagnostic device, we want to deliver a solution for a hospital ward or department that supports clinicians in the diagnosis and treatment and guides them in the clinical protocols during the complete patient's stay. The solution would integrate with many different Philips and non-Philips clinical information systems. Open, standards-based, and interoperable software solutions are required.

One of the challenges in this transition is on clinical pathways and protocols (or workflows) in such solutions. During their stay in a hospital, patients traverse one or more clinical pathways and need to comply with one or more clinical protocols. Compliance with such protocols is often crucial for having the best clinical outcome for the patient.
So, how can we give guidance to the clinical personnel to have the patient traverse the expected clinical pathway/protocol? What flexibility on the protocol implementations does this require (e.g., are they the same for all hospitals)? Ideally, it requires harmonized implementations of these pathways and protocols, so they become sharable between systems, or within a system (i.e., same software interfaces, similar technology choices, etc.). How can we realize such interoperable pathways and protocols, that also work with third party systems?

Philips Research has done an initial assessment of this challenge and has proposed a reference architecture for a generic software solution that loads formalized protocol definitions, executes them, and interacts with other healthcare application through interoperable and standardized interfaces. Now, it is time to verify that this generic solution can solve the challenge described. Many technical and non-technical questions need to be answered.

The work described in this thesis will help our research forward. We have a better understanding on profiling executable BPMN definitions to interoperable FHIR definitions that are commonly used in Healthcare applications. We have a software design that allows working with off-the-shelf BPMN engines of different vendors interchangeably. And finally, we have an end-to-end demonstrator in a Philips inner/open-source repository that helps us build the solution further in a community with Philips and/or external contributors. Thanks a lot Juan van der Heijden, for taking our Research further and helping to make this part of future Philips' solutions.

Bas Bergevoet / Patrick Bonné
September 2021

Preface

This report presents the results of the project 'Standardized software solution for guidance of clinical workflows'. The project was carried out by Juan van der Heijden as a final assignment for the Professional Doctorate in Engineering (PDEng) Software Technology program, offered by Eindhoven University of Technology (TU/e). The PDEng program is a full-time post-master technological design program that leads to a Professional Doctorate in Engineering (PDEng) degree. The program is given under the banner of the 4TU.School for Technological Design, Stan Ackermans Institute, which is a joint initiative of the four universities of technology in the Netherlands.

Patients often traverse through one or multiple clinical pathways during their stay in a hospital. In these pathways, they often encounter many defined clinical workflows. The extent to which these clinical workflows are adhered to is often crucial for a patient's best clinical outcome. Philips is currently researching the feasibility of creating a standardized software solution for guidance of clinical workflows, called the HealthSuite Workflow Capability. The main objective of the project was to create a detailed design and prototype of the HealthSuite Workflow Capability. This detailed design is based on a reference architecture proposed by Philips Research, as a result of prior exploration. The work presented in this report will describe how this objective is attained.

The target audience of this report is people with a technical background, preferably in Computer Science. For non-technical readers, Chapters 1, 2, 3, 4, and 7 are recommended reading material.

Juan van der Heijden

September 2021

Acknowledgements

The work presented in this report would not have been possible without the people around me. I want to express my gratitude towards those people in this section.

I want to thank my company supervisors Bas Bergevoet and Patrick Bonné. You have provided me with supervision, guidance and kept track of my overall progress. At the start of the project, I was inexperienced in many concepts used in the project, such as business process modeling and healthcare standardization. Thank you for the time and effort invested to get me up to speed with these concepts. I have learned from our discussions and the knowledge that you have shared with me. Without your efforts, this work would not have been possible.

I would like to thank my TU/e supervisor Michel Chaudron. I'm grateful for your assistance during the project. Your knowledge about project management and architecture has helped me immensely during this project. I have learned a lot from our interactions.

I would like to thank Philips Research and more specifically the Software Concepts team. The team has supported me with their domain expertise, especially when stuck on technical details. Thanks for being so open, and being easily approachable. Even though direct interactions were hard in times of COVID-19, I was always welcomed for a virtual coffee with open arms.

I would also like to thank the Software Technology PDEng management, coaches, and colleagues. In particular, I would like to thank Desiree van Oorschot and Yanja Dajsuren for their support and guidance during the project, and the program in general.

Finally, I would like to thank my friends and family, your support and encouragement have helped me throughout. A special thanks to my mother, Jolanda, my father, Jos, and my love, Evial, who have always believed in me. Without your faith, support, and strong motivation I would have never pushed my boundaries and reach as far as I did.

Juan van der Heijden September 2021

Executive Summary

During a patient's stay in the hospital, patients traverse through multiple clinical pathways, often having to comply with one or more defined clinical workflows. Information on how a clinical workflow must be executed is often described by large healthcare bodies. These clinical workflows are often locally adapted. Proper adherence to these workflows is often crucial for having the best clinical outcome for the patient.
Philips is currently researching the feasibility of creating a standardized software solution for guidance of clinical workflows, called the HealthSuite Workflow Capability. The HealthSuite Workflow Capability can help healthcare personnel with executing these clinical workflows more accurately. Accurate execution of clinical workflows helps healthcare institutions to improve the quality of their healthcare.
Before the start of the project, a Reference Architecture was defined that presents an overview of the direction Philips wants to take with the HealthSuite Workflow Capability. In this project, we extended this direction by answering the following questions:

- Is the current Reference Architecture feasible to implement?
- What should a detailed design for the realization of the current Reference Architecture look like?
- How should data for clinical workflows be standardized?

This report describes a detailed prototyped design to show the feasibility of the HealthSuite Workflow Capability reference architecture. This includes a suggestion for a standardization approach for clinical workflow data. The prototyped solution is shared within the Philips Innersource Codebase, such that future research can use our prototype as a baseline, and collaboration with other Philips engineering teams is made straightforward.
We designed and prototyped the HealthSuite Workflow Capability based on the BPMN and DMN standards. The proposed detailed design shows how a workflow engine vendor lock-in can be prevented. A modular design is presented that makes the system deployable either on-site or in the cloud.
We recommend to further develop and mature the HealthSuite Workflow Capability by extending the current prototype to also support the CMMN standard. Further, we recommend the exploration of the Clinical Quality Framework to improve aspects of the current prototype.

Glossary

KPI	Key Performance Indicator
EMR	Electronic Medical Record
HL7	Health Level 7
FHIR	Fast Healthcare Interoperability Resources
PDEng	Professional Doctorate in Engineering
ST	Software Technology
TU/e	Eindhoven University of Technology
HSDP	HealthSuite Digital Platform
API	Application Programming Interface
AWS	Amazon Web Services
UML	Unified Modeling Language
OMG	Object Management Group
WHO	World Health Organization
XML	Extensible Markup Language
BPMN	Business Process Model and Notation
CMMN	Case Management Model and Notation
DMN	Decision Model and Notation

List of Tables

List of Figures

Contents

1 Introduction

1.1 Project Context

Philips, a focused leader in the healthcare technology industry, has the purpose of improving people's health and well-being through meaningful innovation. Philips aims to improve 2.5 billion lives per year by 2030, including 400 million in underserved communities. In 2019 Philips invested 1.9 billion euros in R&D. Within Philips Research new technology to improve people's lives is introduced. Software Concepts is a department within Philips Research, consisting of a relatively young, international group of people with a wide area of expertise across the software domain. The department focuses on the use of the latest software technologies for healthcare by making fast prototypes and piloting new software innovations for Philips.
One of the concepts Philips Research has explored are candidate technologies for a generic workflow asset. In the scope of this project, Philips uses the term *workflow* for what is defined as a *dynamic workflow* in the Dynamic Workflow White Paper: *"a sequence of tasks performed by people and systems to accomplish a shared goal. We distinguish a dynamic workflow as a workflow designed and architected for adaptability, flexibility, and extensibility*."[1]
During a patient's stay in the hospital, patients traverse through multiple clinical pathways, often having to comply with one or more defined clinical workflows. Clinical workflows are a specific type of workflow, aimed towards providing healthcare, e.g. a protocol on how to treat sepsis. Proper adherence to these workflows is often crucial for having the best clinical outcome for the patient. Philips' hypothesis is that clinical workflow guidance could help in improving clinical workflow adherence and also in scheduling the tasks of staff in the hospital.[2] Currently, all information regarding clinical workflows is documented in text, or not documented at all. Standardizing and sharing clinical workflows between organizations and businesses has become more common in this day and age. Using a standardized and understandable notation for complex and pliable clinical workflows is a significant challenge.
Philips envisions a standardized software solution for guidance of clinical workflows. This software solution should be able to guide healthcare personnel through these workflows. This idea resulted in the HealthSuite Workflow Capability concept, which will ultimately become part of Philips HealthSuite.[3]

1.1.1 HealthSuite Workflow Capability

The HealthSuite Workflow Capability is a concept presented by Philips in the *Exploration Note HealthSuite Workflow Capability*[2] to tackle the above-stated issues. On a high level, it represents a system capable of giving clinical guidance to healthcare personnel by executing clinical workflows. Figure 1.1 presents a conceptual overview of the purpose of the HealthSuite Workflow capability, additionally indicating what the potential context of the HS Workflow Capability could be.

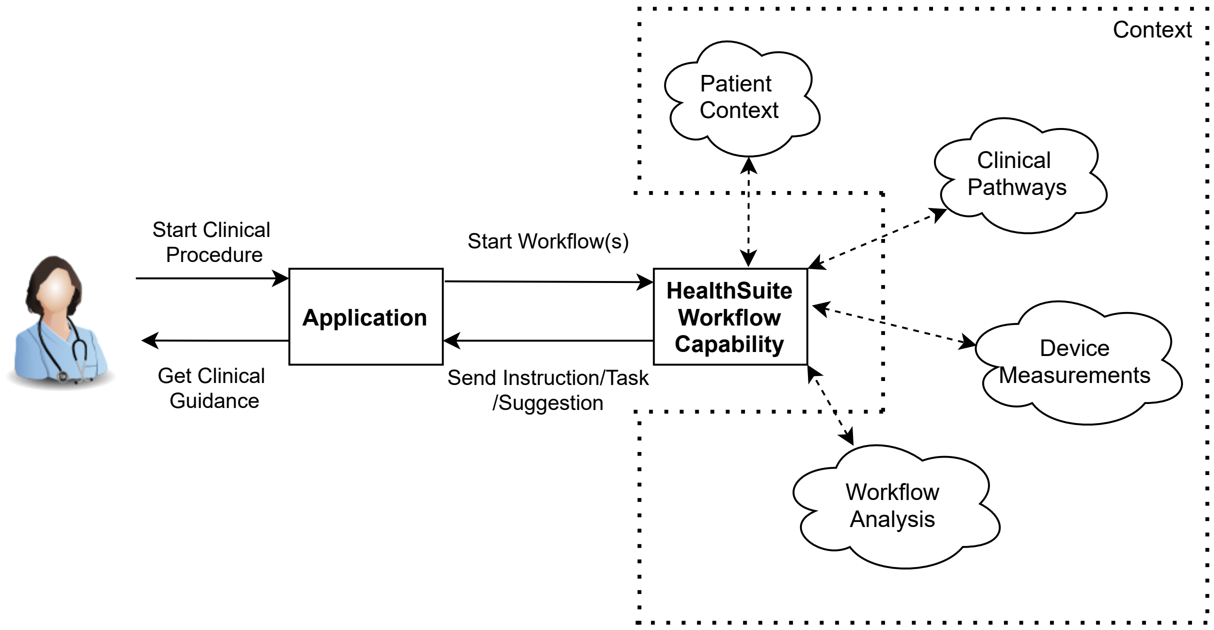


Figure 1.1: Conceptual overview of the purpose and context of the HealthSuite Workflow Capability

Note that Figure 1.1 gives an extremely conceptual overview, the purpose of this Figure is to illustrate how the HealthSuite Workflow Capability would be positioned with respect to its context. A more detailed overview is given in Section 2.4.

1.2 Report Outline

In the remainder of this report, we describe the process and results of this project. Chapter 2 provides an in-depth problem analysis. In Chapter 3 we describe the stakeholders of the project. Chapter 4 captures the requirements gathered during the project. Further, in Chapter 5 the detailed Architecture and Design of the system is discussed. Then, in Chapter 6 we discuss how the system is verified and validated. Finally, in Chapter 7 we conclude the project by giving a summary and recommendations for Future Work.

2 Problem Analysis

To get a proper view of the problem and the context of the problem at sight, we analyze aspects related to the problem. Section 2.1 describes the need for Clinical Workflow Guidance. Section 2.2 describes the need for Data Standardization in healthcare. Section 2.3 describes what knowledge models are and how Philips thinks they should be formalized. Section 2.4 describes the already existing HealthSuite Workflow Capability Reference Architecture made by the Software Concepts team. Section 2.5 describes the scope and goal of this project.

2.1 Clinical Workflow Guidance

In recent years, organizations that pay for healthcare, such as insurance companies, are shifting from a fee-for-service payment model to a value-based care model.[4] A fee-for-service model pays the healthcare organization based on the number of services, like how many patients it treated. A value-based care model pays the healthcare organization based on the quality of their healthcare. This quality is demonstrated using Key Performance Indicators (KPIs), which are measurable variable values. This shift causes healthcare organizations to try and optimize these KPIs. Over the years, the usage of standardized clinical workflows has shown improvement of KPIs and clinical outcomes.[1]

Trivially, these clinical workflows should not only be defined, but also be adhered to. Philips aims to be the middleman between the clinical workflow producer and the clinical workflow actors, by providing a system capable of managing workflows in a healthcare environment. Such a capability can be deployed to support adherence to clinical decisions and actions. However, given a generic workflow asset, it can also support the adherence to operational workflows, e.g. to improve resource allocation. This project will put focus on the clinical workflow use case while considering the operational workflow use case.

2.2 Healthcare Data Standardization

Due to technological advancements in recent years, local care has turned into a network of care, where data is being shared between different caregivers to collectively try and improve healthcare in itself. Additionally, patients nowadays often traverse through different hospitals or organizations. To function effectively, patient data needs to be shared smoothly, which created a demand for systematized collections of data, such as Electronic Medical Records (EMR). Sharing EMRs in different data formats can be problematic for the efficiency of data exchange. This fact made healthcare organizations realize the necessity of data standardization across different healthcare organizations. In 2012, the WHO even stated in a Forum on Health Data Standardization and Interoperability that "Interoperability is widely recognized as essential to achieving the full potential of seamless data exchange using Information and Communication Technologies (ICTs)."[5]

An organization called HL7 (Health Level 7), founded in 1987, recognized this demand for interoperability early. HL7 created an international set of standards for transferring clinical and administrative data between healthcare providers. One of the standards HL7 defined is the Fast Healthcare Interoperability Resources (FHIR) Standard [6]. FHIR is a combination between the HL7 v2 [7], HL7 v3 [8], and CDA [9] standard. FHIR has a strong focus on ease of implementation, making it an attractive choice for software solution providers.

These days, the HL7 FHIR standard is used in healthcare for patient records in multiple hospitals. Although it is uncertain which data standard will eventually be internationally adopted, Philips decided to mainly focus on the HL7 FHIR standard as a means for EMR exchange. Hence, HL7 FHIR is used as a standard in this project for clinical data exchange.

2.3 Clinical Knowledge Models

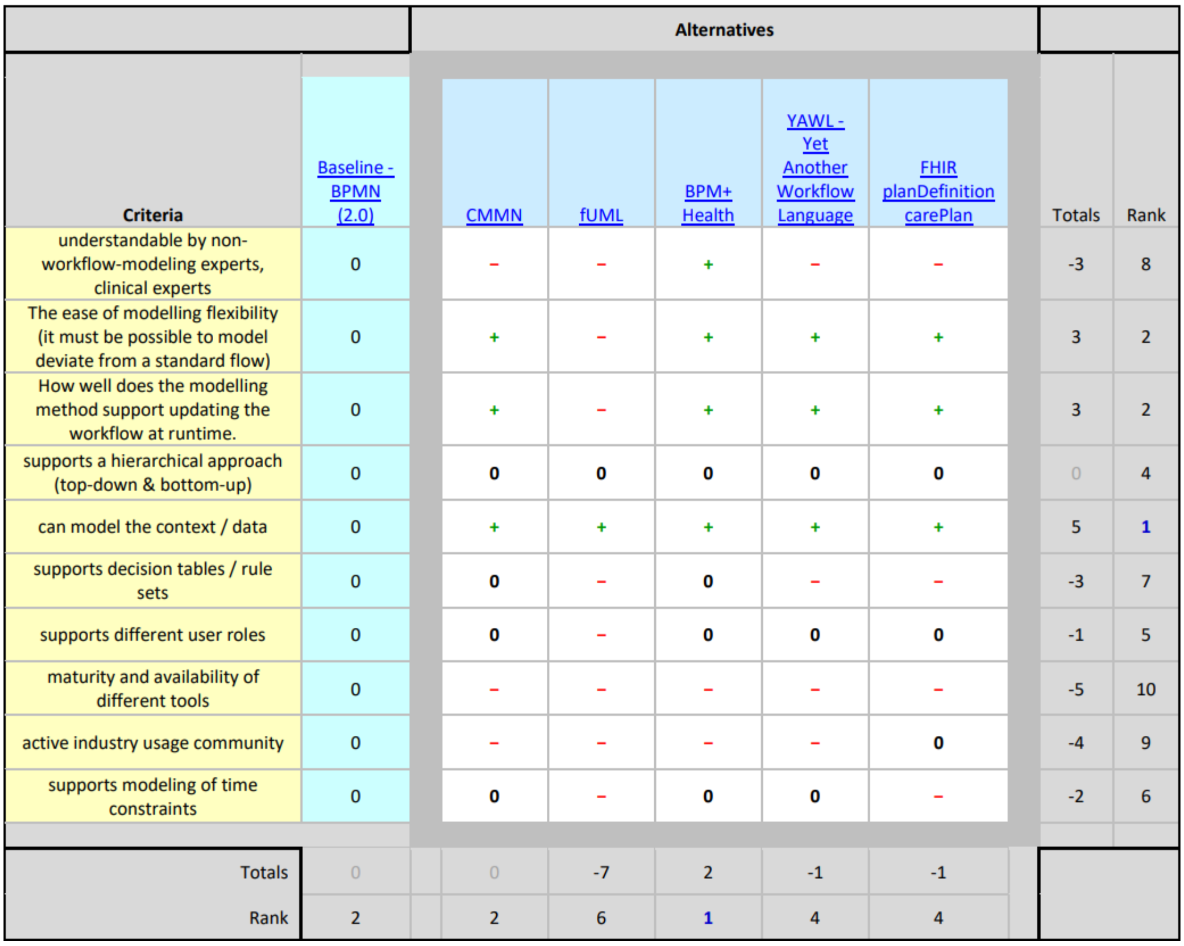
From the findings in Section 2.1 and 2.2, we can conclude that a software solution for workflow guidance in healthcare can be beneficial. The question, however, is how to formalize these workflows, such that they can be interpreted and executed. Philips has done thorough research on different methods and tools to express workflows. Figure 2.1 shows the results from their *Exploration Note HealthSuite Workflow capability* [2] in a Pugh matrix. BPMN 2.0 is used as a baseline since it is the most well-known workflow modeling language. As alternatives, CMMN, fUML, BPM+ Health, YAWL, and FHIR are used. By recommendation of Philips, we continued exploring BPM+ health as a candidate for expressing Clinical Knowledge Models.

2.3.1 BPM+ Health

BPM+ Health is a community of practice with the goal of improving national and international health. Their mission is to: [10]

- Increase the adoption of evidence-based medical best practices.
- Promote the use of industry standards to allow practitioners to share and adopt clinical pathways, guidelines, and other healthcare knowledge.
- Develop best practices for modeling and sharing clinical pathways, clinical guidelines, and other healthcare knowledge.
- Foster collaboration and alignment with other standards development organizations and relevant work in the healthcare industry.

In 2020 the community released a *Field Guide to Shareable Clinical Pathways*. In this release, they proposed a standard for representing clinical pathways. They focus on a model-based approach of representing clinical workflows, utilizing the already existing standards for business process modeling



		Alternatives						
Criteria	Baseline - BPMN (2.0)	CMMN	fUML	BPM+ Health	YAWL - Yet Another Workflow Language	FHIR planDefinition carePlan	Totals	Rank
understandable by non-workflow-modeling experts, clinical experts	0	−	−	+	−	−	-3	8
The ease of modelling flexibility (it must be possible to model deviate from a standard flow)	0	+	−	+	+	+	3	2
How well does the modelling method support updating the workflow at runtime.	0	+	−	+	+	+	3	2
supports a hierarchical approach (top-down & bottom-up)	0	**0**	**0**	**0**	**0**	**0**	0	4
can model the context / data	0	+	+	+	+	+	5	1
supports decision tables / rule sets	0	**0**	−	**0**	−	−	-3	7
supports different user roles	0	**0**	−	**0**	**0**	**0**	-1	5
maturity and availability of different tools	0	−	−	−	−	−	-5	10
active industry usage community	0	−	−	−	−	**0**	-4	9
supports modeling of time constraints	0	**0**	−	**0**	**0**	−	-2	6
Totals	0	0	-7	2	-1	-1		
Rank	2	2	6	1	4	4		

Figure 2.1: Pugh matrix on modelling languages

and decision modeling. BPM+ Health suggests representing clinical workflows using the following standards:

- Business Process Model and Notation (BPMN) standard [11]
- Case Management Model and Notation (CMMN) standard [12]
- Decision Model and Notation (DMN) standard [13]

The Business Process Model and Notation (BPMN) standard was made with the main goal in mind: to provide a notation that is readily understandable by all business users. It is well known and one of the most used notations, which is why in BPM+ Health it is the standard choice for workflows. However, BPMN has some infirmities in the healthcare domain compared to the other standards mentioned above, e.g. it's hard to deviate from the standard flow in BPMN because of its deterministic sequence nature. The BPM+ Health community recognized the need for more flexibility in the healthcare domain and proposed to add the Case Management Model and Notation (CMMN) standard to complement the BPMN standard. The CMMN standard uses case models that are dependent on context information and don't follow a strict flow, unlike the BPMN standard. CMMN is often criticized for being complex for workers. The flexibility of CMMN bridges the gap that BPMN lacks. The Decision Model and Notation (DMN) is used for decision support in BPMN and CMMN. It consists of one or more tables that are easy to understand by clinical personnel. In this way, the decision tables and flow modeling are clearly separated. Therefore, BPM+ Health proposes to integrate the BPMN, CMMN, and DMN standards in a single model.
Purely looking at the BPM+ Health definition, it would seem like the perfect candidate to capture clinical knowledge. However, a clear disadvantage of using BPM+ Health over BPMN is the lack of maturity and availability of tools that combine the BPMN, CMMN, and DMN standards. BPMN 2.0 has mature engines ranging back from open source Orchestra [14] engine from 2012 to the more recently developed platforms like jBPM [15] and Camunda [16]. However, the combination of the three models needs either a new engine that supports this combination or a set of engines that have some form of connection between them.

In the remainder of the report, we will often refer to Clinical Knowledge Models as simply Knowledge Models. Knowledge Models will be defined as: "A collection of information that uses a combination of workflow standards, which together express a clinical workflow". Based on the findings above, we envision a knowledge model based on at least the BPMN, CMMN, and DMN standards. When we refer to the Knowledge Model, it can be assumed it includes these standards.

2.4 HealthSuite Workflow Capability Reference Architecture

The HealthSuite Workflow Capability Reference Architecture is a proposed design for the standardized software solution for guidance of clinical workflows. This Reference Architecture is described in the Exploration note on HS Workflow Capability.[2] This section elaborates on the proposed design including diagrams. The Reference Architecture is presented Figure 2.2. We can denote four main segments:

- HealthSuite Workflow Capability

- FHIR Store
- Standardized actions, data, and triggers interface
- Software application

Each of these segments is briefly described based on function and structure.

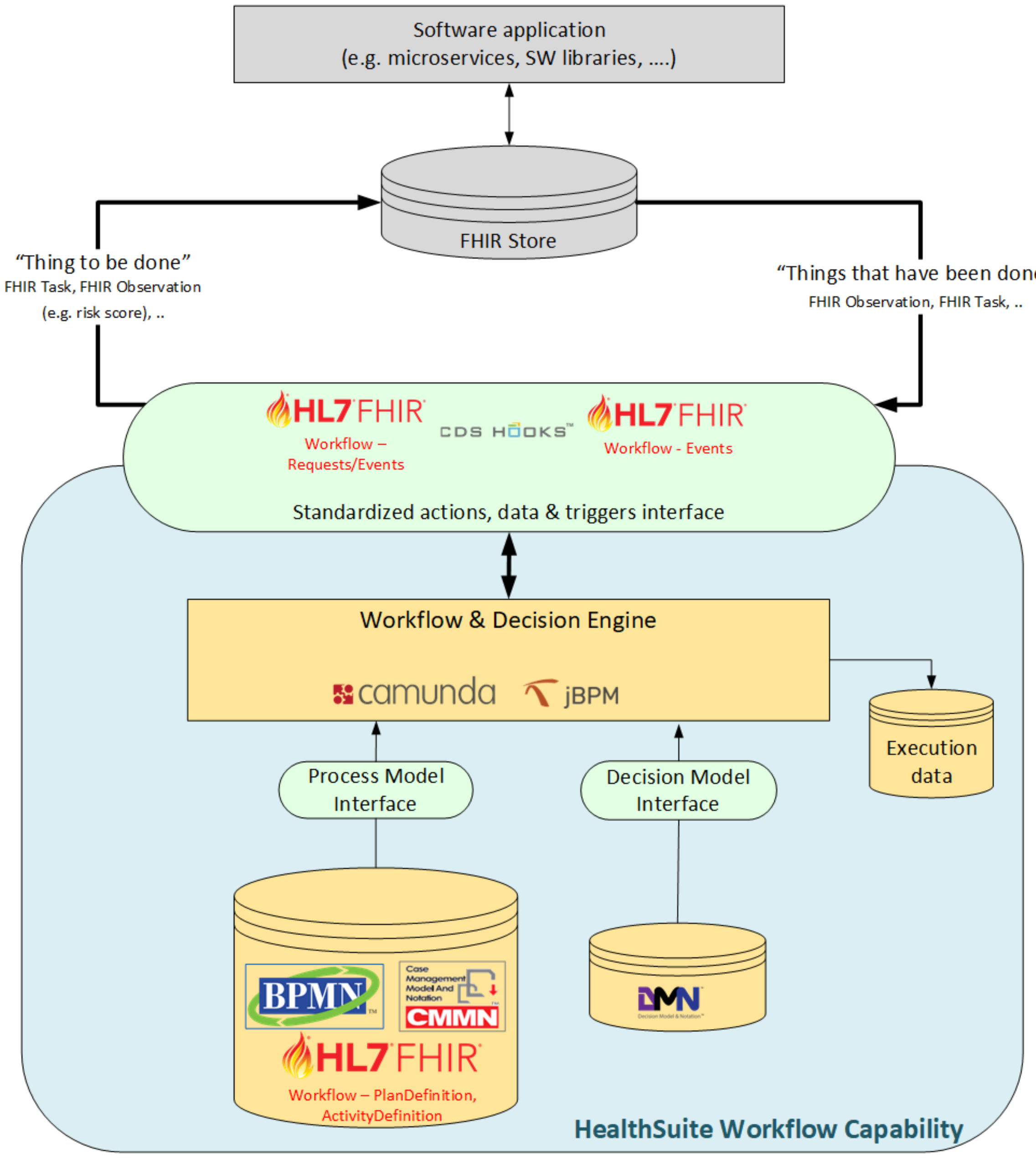

Software application
(e.g. microservices, SW libraries, ...)
FHIR Store
"Thing to be done"
FHIR Task, FHIR Observation
(e.g. risk score), ..
"Things that have been done"
FHIR Observation, FHIR Task, ..
HL7 FHIR
Workflow – Requests/Events
CDS HOOKS
HL7 FHIR
Workflow - Events
Standardized actions, data & triggers interface
Workflow & Decision Engine
camunda
jBPM
Execution data
Process Model Interface
Decision Model Interface
BPMN
Case Management Model And Notation
CMMN
HL7 FHIR
Workflow – PlanDefinition, ActivityDefinition
DMN
HealthSuite Workflow Capability

Figure 2.2: Reference architecture for the HS Workflow Capability

2.4.1 HealthSuite Workflow Capability

The HealthSuite Workflow Capability has the function of running a Knowledge Model. In the reference architecture, the suggestion for the BPM+ Health standard has been made as described in Section 2.3. This standard includes Process Models denoted in BPMN and/or CMMN and Decision models denoted in DMN. These models are imported into a Workflow and Decision Engine through an interface. The choice of technology for the Workflow and Decision Engine is still open for research. The function of the Engine is to execute the imported models. Data needed by the models or produced by the models are communicated through the Standardized actions, data, and triggers interface. Additionally, the Engine can produce Execution data which is recorded information about what happened during the execution of a model, e.g. the exact path traversed for a single patient.

2.4.2 Standardized actions, data, and triggers interface

The standardized actions, data, and triggers interface needs to allow communication between the HealthSuite Workflow Capability and applications interacting with the HealthSuite Workflow Capability. For this, a standardized method of communication needs to be used. To accommodate this, the design decision to use HL7 FHIR is used for all communication from and to the HS Workflow Capability as described in Section 2.2.

2.4.3 FHIR Store

The FHIR Store is a data store with a predefined and known structure. In the HealthSuite Workflow Capability Reference Architecture, it acts as an interface between the HS Workflow Capability and the HS Workflow Capability context. HL7 FHIR is widely used within Philips systems and non-Philips systems, making the integration of already existing systems effortless.

2.4.4 Software application

Software applications can communicate with the FHIR Store to use and provide information used/generated by the HS Workflow Capability. Specifics about the software application are kept purposefully indistinct, meaning there could be multiple software applications serving different use cases.

2.5 Scope and Goal

This project aims to create a detailed software design and prototype of the HealthSuite Workflow Capability, based on the high-level conceptual reference architecture presented in Section 2.2. To achieve this, relevant standards that contribute to the workflow solutions are studied. These results are demonstrated and presented to various stakeholders within Philips. This project has limited resources; Hence we focus on the two research questions presented in Table 2.1.

Table 2.1: Research Questions

Question ID	**Research Question**
RQ-01	Based on the high-level conceptual HealthSuite Workflow Capability Reference Architecture, what should a detailed software design for the HealthSuite Workflow Capability look like?
RQ-02	How can the elements of the BPMN standard be mapped to the HL7 FHIR standard, such that software applications can interact with workflows through the FHIR Store?

The previous research questions led to several objectives listed in Table 2.2

Table 2.2: Project Objectives

Objective ID	**Objective Description**
OB-01	Capture the primary requirements high-level requirements, based on use cases of the system.
OB-02	Create a detailed software design for the HealthSuite Workflow Capability, following the HealthSuite Workflow Capability Reference Architecture.
OB-03	Create a prototype based on the architecture and design defined as a result of **OB-02**
OB-04	Describe a method on how information and interaction with a Knowledge Model can be realized through the HL7 FHIR Standard
OB-05	Provide documentation for the design and prototype to support future development and research by Philips.

3 Stakeholder Analysis

In this chapter, the main stakeholders of the project are identified. The stakeholders are categorized into direct and indirect stakeholders. The direct stakeholders have a direct influence on the execution and/or supervision of the project. Indirect stakeholders are stakeholders not directly involved in the project. However, they need to be considered for the design of the system.

3.1 Direct Stakeholders

3.1.1 Philips Stakeholders

Bas Bergevoet and Patrick Bonné

Role: Bas Bergevoet and Patrick Bonné are the project supervisors from Philips. They are the interface regarding all demands from Philips and act as the product owners. They provide valuable feedback regarding the product and findings. They also assist in the communication with other stakeholders or technology experts within Philips.
Concerns:

- Detailed design proposal on the HealthSuite Workflow Capability
- New findings regarding the HealthSuite Workflow Capability
- Matured and extended demonstrator of the HealthSuite Workflow Capability in Philips inner-source

3.1.2 Eindhoven University of Technology Stakeholders

Michel Chaudron

Role: Michel Chaudron is the project supervisor from the TU/e. He is the interface regarding the demands from the TU/e. Michel helps with the creation of the architecture and documentation by having regular meetings.
Concerns:

- Formal architecture of the system, defined in the Unified Modeling Language (UML)
- Successful project delivery
- High-quality final report

Yanja Dajsuren

Role: Yanja Dajsuren is the program director of the Professional Doctorate in Engineering - Software Technology (PDEng ST) program. She is responsible for the whole PDEng ST performance. Yanja sets the expectation from the TU/e and assesses the results of the project.
Concerns:

- Quality of the project results
- Future collaboration with Philips
- Graduation of all PDEng ST Trainees

3.1.3 Indirect Stakeholders

Caregivers

Role: One group of the eventual users of the system. This project mainly focuses on the use case for caregivers, so they are an important indirect stakeholder.
Concerns:

- Usability of the guidance application
- Accuracy of the suggested actions of the system
- Flexibility of the system, when an action is taken which deviates from the original protocol

Healthcare Institutions

Role: Healthcare Institutions will be the target customers for the systems. They are supposed to deploy the system within their hospitals. They are mostly focused on the financial side and the effectiveness of the system.
Concerns:

- Cost of using the HealthSuite Workflow Capability
- Improvement on KPIs while using the HealthSuite Workflow Capability
- Where the HealthSuite Workflow Capability will be deployed

Philips Device Manufacturers

Role: Philips creates many solutions for healthcare, from X-Ray to Patient Monitoring devices[17]. The HealthSuite Workflow Capability should be able to interact with other systems, for example when a workflow needs a certain measurement or wants to instruct another device.
Concerns:

- Compliance of the system with the Reference Architecture set by the Philips Chief Architect Office.

- Standardization of the input and output data of the system.

Knowledge Model Producers

Role: At Philips, technology to shape the future of Healthcare is made. Philips does not advise Healthcare givers on the clinical workflows that should be used. For that, we have medical publishers such as the World Health Organization [18], Elsevier [19], and Thieme Medical Publishers [20]. Sometimes, clinical workflows are even produced or adjusted by local hospitals.
Concerns:

- Proper standardization of the Knowledge Model Definition
- Understandability of the Knowledge Models by nontechnical Healthcare experts.

4 Requirements and Use Cases

This chapter describes the requirements and use cases for the HealthSuite Workflow Capability detailed design and prototype. These requirements were defined during the project and are scoped towards the project objective. Section 4.1 discusses all relevant actors in the system. We list the system requirements in Section 4.2. To give context to the system, we describe system scenarios in Section 4.3. Finally, Section 4.4 describes the use cases of the prototyped solution presented in this thesis.

4.1 Actors

We will discuss the relevant actors with respect to the scope and use case of the system. Some users of the system are already listed with their interests in Section 3.1.3. This section elaborates on the role of the direct actors in the system. Table 4.1 represents all relevant actors of the system with their respective roles with respect to the system.

Table 4.1: System Actors

Actor Name	Role
Caregiver	The Caregiver uses the system to get clinical guidance on the treatment of a patient. They are interested in the ease of use of the system and the available information.
Clinical Expert	The clinical expert performs tests and analysis concerning a certain patient. The data produced by these tests and analysis is later used by workflows, and therefore has to be communicated to the system.
Workflow Producer	A workflow producer creates a knowledge model based on clinical procedures. Their main concern is the standardization used in the system and the deployability of the knowledge model.

4.2 System Requirements

In the Exploration Note HealthSuite Workflow capability [2] a set of requirements is defined collected from various discussions with contacts within Philips. The prototyped design solution presented in this thesis will consist of a subset of these requirements. Further, some of the requirements in the exploration note will be further refined. The full list of requirements for the HealthSuite Workflow Capability, as defined by Philips in the exploration note, is presented in Appendix B. The non-functional requirements for the HealthSuite Workflow Capability design and prototype are listed in Table 4.2, the functional requirements are listed in Table 4.3.

Table 4.2: Non-functional Requirements

Requirement ID	Requirement	MoSCoW
NF-01	Workflows shall be easily understandable by users of the system	M
NF-02	The system shall support customers to personalize the defined workflows.	S
NF-03	The system shall have no hard dependencies on a Workflow Engine vendor, i.e. no vendor lock-in.	S
NF-03a	The system shall have a modular design that allows the Workflow Engine to be replaced.	S
NF-04	The system shall be easy to integrate with other systems, by using a standardized communication protocol.	M
NF-05	The system shall be shareable in a software community, such that other engineers in the community can easily add to and extend the solution.	M
NF-06	The system shall separate responsibilities between the solution supplier and the protocol workflow creator.	M

Table 4.3: Functional Requirements

Requirement ID	Requirement	MoSCoW
F-01	The system shall be able to support BPMN models, such that strictly procedural parts of the workflow are accommodated for.	M
F-02	The system shall be able to support CMMN models, such that unstructured, dynamic planning in the workflow is accommodated for.	C
F-03	The system shall be able to support DMN models to support decision tables.	S
F-04	The system shall allow XML input for the OMG defined model notations.	M
F-04a	The system shall allow XML input for the BPMN standard.	M
F-04b	The system shall allow XML input for the CMMN standard.	C
F-04c	The system shall allow XML input for the DMN standard.	S
F-05	The system shall communicate data to the system's context through the FHIR standard.	M
F-06	The system shall allow BPMN/CMMN/DMN models to use clinical data, typically stored in FHIR Resources.	S
F-07	The system shall allow Knowledge Models to be deployed to the system.	M
F-07a	The system shall allow deployed Knowledge Models to be updated in the system.	S
F-08	The system shall use a uniform, programming-language agnostic, communication protocol for interaction with the Workflow Engine.	S

F-9	The system shall have a readme document describing the dependencies, building process and functionality of the system.	S
F-10	The system shall be shared within the Philips Innersource GitHub environment.	M
F-11	The system shall have a standalone workflow deployment application.	M
F-12	The system shall have a standalone guidance application.	M
F-12a	The standalone guidance application shall exclusively communicate with the system through FHIR data.	S
F-13	The standalone guidance application shall show a list of outstanding tasks to be done by the caregiver.	M
F-14	The system shall allow the caregiver to update a user task status by the means of a standalone guidance application.	M

The non-functional requirements in Table 4.2 are satisfied by the functional requirements in Table 4.3. To show the traceability between these requirements, a mapping is presented in Table 4.4.

Table 4.4: Requirements Mapping

Non-functional ID	Functional ID
NF-01	**F-01, F-02, F-03, F-04, F-13, F-14**
NF-02	**F-07(a)**
NF-03	**F-06, F-08**
NF-04	**F-05, F-06, F-12a**
NF-05	**F-9, F-10**
NF-06	**F-11, F-12**

4.3 Scenarios

Prior to the design and implementation, we had a brainstorming session on potential Use Cases of the system. These are captured in scenarios listed below. These scenarios can help building the bigger picture of future and indirect use cases of the system. These scenarios envision a more mature version of the HealthSuite Workflow Capability and potential usages.

4.3.1 Scenario 1 — Clinical Guidance

User: Caregiver
Scenario: A caregiver gives personalized care to a patient for which a workflow is currently running. The workflow evaluates an expression using prediction scores based on blood measurements made by external devices. The result of the evaluation shows that checking the blood pressure of the patient is important for the next step in the workflow. As a result, the system shows the blood pressure values on the screen.
Questions:

- What decides these prediction scores?
- Where is the data for these prediction scores coming from and stored? (knowledge models?)
- What does "displaying values" mean? Simply a notification with these values might be important or is the screen occupied while displaying predictions?

4.3.2 Scenario 2 — Workflow Creation

User: World Health Organization
Scenario: WHO comes up with a new protocol to treat a patient with Corona. They describe a clinical workflow for caregivers on how to deal with a Corona case. A middleman offers, as a service, to translate this clinical workflow to a knowledge model which can be imported into the HealthSuite Workflow Capability system. As a result, hospitals can use the Workflow Capability to execute the new WHO clinical workflow.
Questions:

- What does such a protocol from the WHO now look like?
- How to translate the WHO clinical workflow to a knowledge model interpretable by the HealthSuite Workflow Capability?
- Who translates the WHO clinical workflow to an interpretable knowledge model?
- Which aspects from the clinical workflow belong in a knowledge model, and which don't?
- How can we make a standard for such knowledge models?

4.3.3 Scenario 3 — Workflow Modification

User: Healthcare Institution
Scenario: The WHO came up with a new protocol to treat a patient with Corona, which is translated to a knowledge model interpreted by the HealthSuite Workflow Capability system. A hospital that is to use this knowledge model wants to slightly alter it for the sake of personal preference with caregiving personnel.
Questions:

- Who will alter the knowledge model such that it is personalized for the hospital?
- Should a knowledge model be fully alterable or should there be limits set?
- How to identify limits in knowledge models, in the case they exist?

4.4 Use Cases

Figure 4.1 presents the use cases of the current HealthSuite Workflow Capability prototype. The use cases are adjusted to the scope of the project, hence they are more narrow than the scenarios sketched in 4.3.

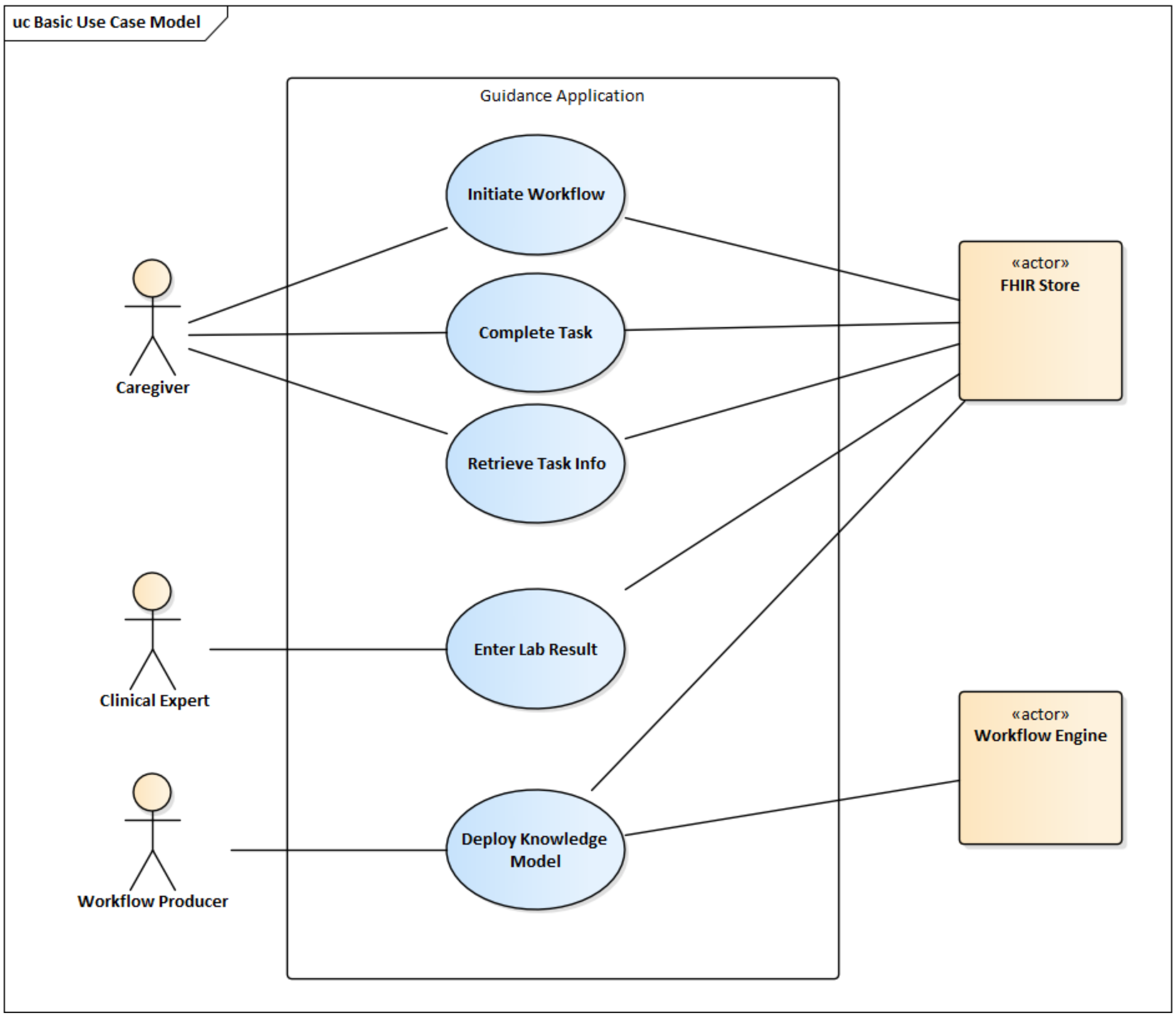


Figure 4.1: Use Case Diagram of the HealthSuite Workflow Capability

5 Architecture and Design

In this chapter, we discuss the detailed Architecture and Design of the HealthSuite Workflow Capability system. In Section 5.1, we discuss the Software Architecture method used. In Section 5.2, we describe a method to characterize the functionality of certain packages and classes. Section 5.3 describes a method for mapping the BPMN standard to the FHIR standard. In Section 5.4, 5.5, 5.6, and 5.7, we discuss the logical, process, physical, and development architectural views respectively. Section 5.8 describes a query language used for data exchange in the HealthSuite Workflow Capability.

5.1 Kruchten 4+1 view model

Since this is a large project, it is helpful to look at academic and industry standards for describing the architecture and design. A well-known and appreciated architecture framework is the "4+1 view model" developed by Philippe Kruchten [21]. The 4+1 view model has been used successfully in many large projects. This model works well since it takes multiple perspectives or *views* on the system. These different views are all evolved from specific scenarios, which we often write as use cases. The 4+1 view model is presented in Figure 5.1.

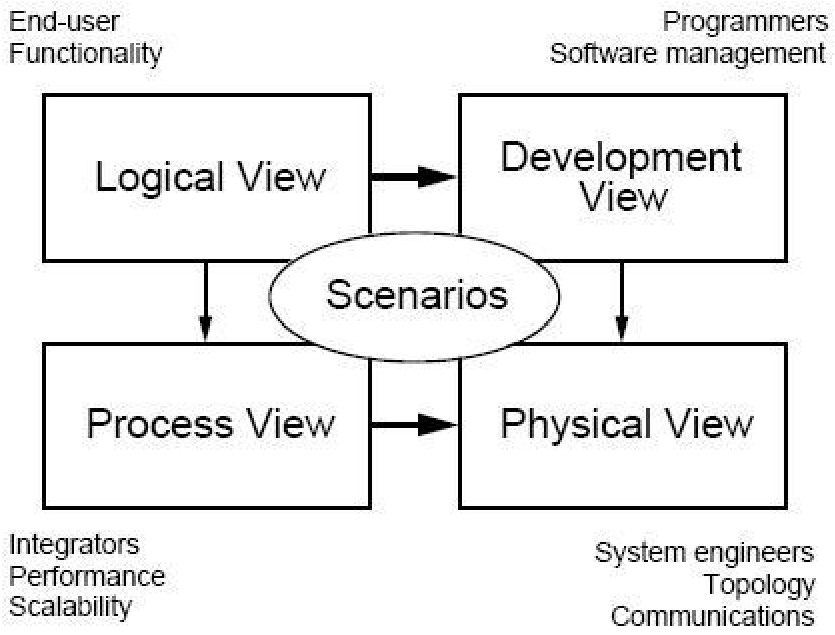


Figure 5.1: The 4+1 view model

The model proposed by Kruchten is made up of five main views:

- The *logical* view, which takes the end-user and functionality into account. In the UML notation [22] the logical view often consists of class diagrams and state diagrams [23].
- The *process* view, which captures the concurrency and synchronization aspects of the design. It often denotes flows and sequences of actions in a system. In the UML notation, the process view often consists of sequence diagrams, communication diagrams, and activity diagrams [23].
- The *physical* view, which describes the mapping(s) of the software onto the hardware and reflects its distributed aspect. It gives the point of view of a system engineer. In the UML notation, the physical view often consists of deployment diagrams [23].
- The *development* view, which describes the static organization of the software in its development environment. It gives a more general view of the system and its connections. In the UML notation, the development view often consists of package diagrams [23].
- The *scenarios*, which describe the interaction of an end-user with the system. It also gives different viewpoints from different stakeholders. To describe the scenarios, use cases are used. These use cases have already been presented in Section 4.4.

5.2 Characterizing Functionality

To convey different functionalities of packages and classes we use the Characterizing Classes methodology proposed by Rebecca J. Wirfs-Brock [24]. UML Diagrams are often used in multiple levels of abstraction, they are used to give a high-level system overview, but also to denote low-level connections between elements in the system. To use UML Diagrams to give a high-level system overview, details have to be abstracted away. This often results in having limited room to describe the functionality or behavior of a class or package. Wirfs-Brock recognized this issue and came up with role stereotypes. Role stereotypes are purposeful oversimplifications that indicate the functionality of a package or class. The models in this section will be color-coded based on it's role stereotype if this makes the functionality more clear. The meaning of the color-coding can be found in a legend presented in the model.

5.3 FHIR Mapping

The HealthSuite Workflow Capability Reference Architecture defined in Figure 2.2 has the FHIR Store as its only interface to external applications. These applications need to know certain information about the Clinical Knowledge Model, such as the status of a BPMN model, or information about a certain Task that has to be executed. Since applications can only access data through the FHIR Store, information about the Clinical Knowledge Model needs to be stored in the FHIR Store.
The HL7 FHIR standard does not support the BPMN, CMMN, and DMN standards. However, they do have their own definition of a Workflow [25], consisting of multiple FHIR Resources. Therefore, we must find a way to denote information about the Knowledge Model, that is needed by the application, in the FHIR Workflow Standard.
This section proposes a FHIR Mapping, which maps elements in the Knowledge Model to the FHIR

Workflow Standard. Data that is mapped to the FHIR Store can be used by applications, therefore we have to choose which data from the Knowledge Model we want to map. In this prototype we chose to map the following data from the Knowledge Model to FHIR:

- General information about the workflow definition, e.g. the workflow name and description
- Information about the instance of a workflow, e.g. which patient is the workflow instantiated for
- Information about tasks, needed by the user of the system
- Information about the status of a task

This means, execution data, such as transitions, decision tables, and expressions, are not included in this FHIR Mapping. The system implemented in this project uses solely BPMN and DMN. Since decision tables don't give any information included in the list above, we don't map the DMN models. However, we have mapped essential elements from the BPMN standard to the HL7 FHIR standard. Unfortunately, time didn't allow for us to map the complete BPMN specification to HL7 FHIR.

In the FHIR Mapping, we make a distinction between the FHIR definition and the FHIR instance of a process. A definition captures the information given by the BPMN model. An instance is an occurrence of the definition, which can have a certain state. In the FHIR Mapping, we assume the standard BPMN 2.0 XML definition[11] proposed by the Object Management Group.

5.3.1 Business Process Model

The root tag of a XML BPMN 2.0 definition is the *bpmn:process* tag. It contains the primary information of the BPMN model, like the id and name of the process. A *PlanDefinition* in HL7 FHIR allows for the definition of various types of plans as a sharable, consumable, and executable artifacts. A *PlanDefinition* has multiple informational fields and can contain actions. This makes a *PlanDefinition* a good fit for mapping the root BPMN structure.

Table 5.1: Business Process Model Mapping

BPMN XML Tag	**BPMN Attribute**	**FHIR Definition**	**FHIR Instance Request**
bpmn:process		PlanDefinition	CarePlan
	id	PlanDefinition.identifier.value	CarePlan.identifier.value
	name	PlanDefinition.name	CarePlan.name
			CarePlan.status = "draft"
			CarePlan.category = " Workflowcapability"
			CarePlan.intent = "plan"
			CarePlan.subject.reference = Patient Ref

Applying a PlanDefinition

HL7 FHIR allows for applying a *PlanDefinition*. This transforms the definition into an instance. In HL7 FHIR an instance of a *PlanDefinition* is called a *CarePlan*. In our mapping, we transfer the values from the definition to the instance and add additional information of the instance.
A full mapping of the Business Process Model is presented in Table 5.1. Note that some FHIR Instance Request elements have no BPMN counterpart, this is because this is information about the state of a workflow instance, which is not captured in the workflow definition.

5.3.2 User Tasks

User Tasks are a core element in BPMN. User tasks must be completed by a user on the application side, so it is important this is properly mapped. For the definition of a *User Task*, we append it to the list of actions in the *PlanDefinition* defined above. A full mapping of the User Tasks is presented in Table 5.2.

Table 5.2: User Task Mapping

BPMN XML Tag	**BPMN Attribute**	**FHIR Definition**	**FHIR Instance Request**
bpmn:userTask		PlanDefinition.action	Task
	id	PlanDefinition.action.id	Task.identifier.value
	name	PlanDefinition.action.name	
			Task.status = "ready"
			Task.intent = "unknown"

5.3.3 Receive Tasks

In the BPMN definition, a receive task indicates that the process has to wait for an incoming message before continuing. When the message is received the task is completed. In the Workflow Capability Prototype we use a combination of the Receive Task with an attached Data Object to denote we need external data to further execute the workflow. The data arrives together with the message indicating that the workflow can continue. How exactly this data is retrieved is explained in Section 5.5.3.
Information about the Receive Task definition is only relevant for interaction with the FHIR Store to fetch data, and not for users of the system. Therefore, Receive Tasks are not mapped to FHIR Resources.

5.4 Logical view

The logical view of the system describes the high-level system functionality. Figure 5.2 presents an overview of all packages in the system. The system is expressed in four different modules: the **Application Module**, **Data Module**, **Control Module**, and the **Execution Module**. The users of the system interact exclusively with the Application Module of the system. In the next sections, each of these modules will be further described.

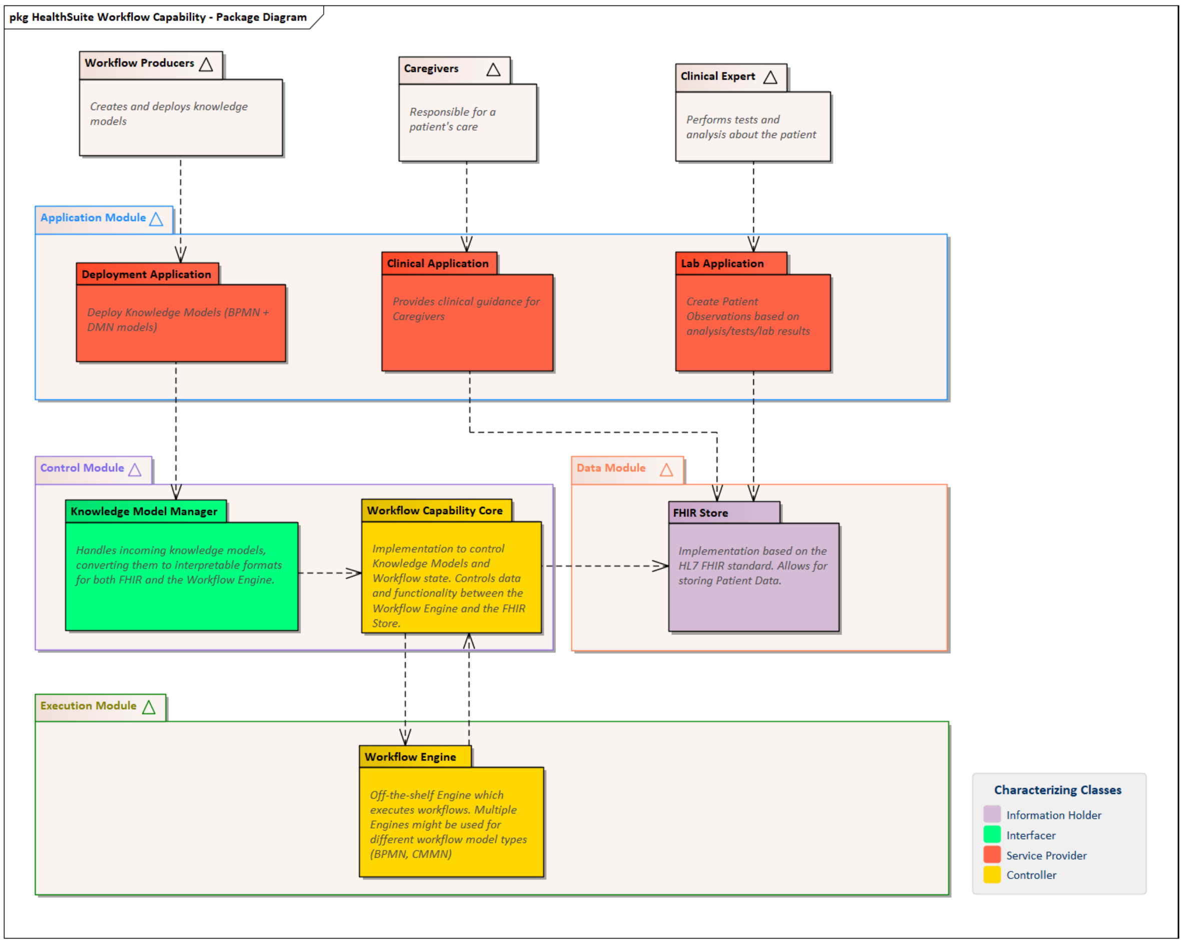

pkg HealthSuite Workflow Capability - Package Diagram
Workflow Producers
Creates and deploys knowledge models
Caregivers
Responsible for a patient's care
Clinical Expert
Performs tests and analysis about the patient
Application Module
Deployment Application
Deploy Knowledge Models (BPMN + DMN models)
Clinical Application
Provides clinical guidance for Caregivers
Lab Application
Create Patient Observations based on analysis/tests/lab results
Control Module
Knowledge Model Manager
Handles incoming knowledge models, converting them to interpretable formats for both FHIR and the Workflow Engine.
Workflow Capability Core
Implementation to control Knowledge Models and Workflow state. Controls data and functionality between the Workflow Engine and the FHIR Store.
Data Module
FHIR Store
Implementation based on the HL7 FHIR standard. Allows for storing Patient Data.
Execution Module
Workflow Engine
Off-the-shelf Engine which executes workflows. Multiple Engines might be used for different workflow model types (BPMN, CMMN)
Characterizing Classes
Information Holder
Interfacer
Service Provider
Controller

Figure 5.2: Logical view - Package Diagram

5.4.1 Application Module

The Application Module provides direct services to users of the system, providing a User Interface. The Application Module interacts with the Data Module for retrieving data and interacting with the workflows, which means that the only interaction coming from an application is through data complying with the HL7 FHIR data standard. Further, the Application Module can interact with the Control Module to deploy Knowledge Models to the system. The Application Module allows for an arbitrary number of applications, for which each of them can have its own use case. In this prototype, we decided to go with three applications, a Deployment Application, a Lab Application, and a Guidance Application. The purpose and functionality of these applications are described below.

Design Decision: The reason for exclusively exposing the FHIR Store has to do with requirement **NF-04**. As mentioned before, the HL7 FHIR standard is already widely used within Philips and non-Philips systems, making integration more effortless. This is also indicated by the Reference Architecture in Figure 2.2. Prior research from Philips has shown that many applications are moving towards the HL7 FHIR standard for data exchange. This means that new applications which want to interact with the HealthSuite Workflow Capability can do so by natively interacting with the FHIR Store.

Deployment Application

The Deployment Application has the purpose of taking Knowledge Module elements like input and deploying them to the HealthSuite Workflow Capability. It's the only application in the Application Module that doesn't interact with the Data Module, but directly to the Control Module. In the current prototype implementation of the system, the Deployment Application takes in raw BPMN and DMN models expressed in the Extensible Markup Language (XML) standard.

Lab Application

The Lab Application is used by Clinical Experts to insert data about a certain Patient in the FHIR Store. The input data is a measurement or simple assertion, which the application turns into a FHIR Observation [26] Object.

Guidance Application

The Guidance Application is used to interact with workflows deployed in the system. According to the Use Cases defined in section 4.4, a caregiver should be able to initiate workflows, complete tasks, and retrieve task info. How these functions are technically realized is described in section 5.7

5.4.2 Data Module

The Data Module is responsible for the access to stored clinical data. As recommended by Philips, this is realized by using an implementation of the HL7 FHIR standard [6], called a FHIR Store. As follows, all data that complies with the HL7 FHIR Standard can be stored and accessed through the FHIR Store. This includes Patient data, Clinical data, and also Workflow data. This data will be used as a means of communication between the Application Module and the Control Module.

5.4.3 Control Module

The Control Module contains the logic to handle Knowledge Models and controls the access to the Workflow Engine and FHIR Store. It consists of two parts: the **Knowledge Module Manager** and the **Workflow Capability Core**. Their purpose in the Control Module is described below.

Knowledge Model Manager

The Knowledge Model Manager handles incoming Knowledge Models. Its concrete functionality is converting incoming XML model definitions of BPMN and DMN models to interpretable formats by the FHIR store and the specific Workflow Engine used. The latter is necessary because Workflow Engines can occasionally introduce specific XML tags extending the BPMN standard, the Knowledge Model Manager makes sure the incoming generic BPMN model is interpretable by the specific Workflow Engine. Functionality is split into one controller and three service providers; A service for creating FHIR Objects, a service for extracting workflow elements, and a service for manipulating the XML of the workflow such that the Workflow Engine can interpret it. The Class Diagram for the Knowledge Model Manager package is presented in Figure 5.3.

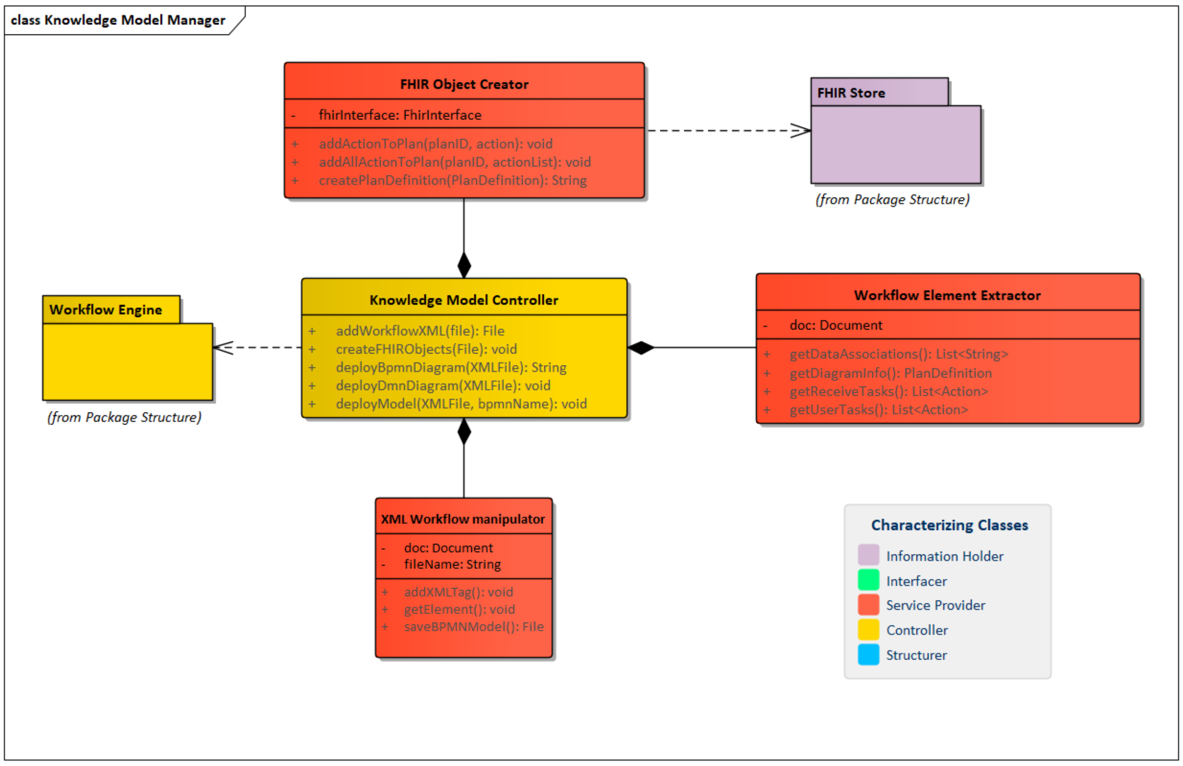

class Knowledge Model Manager
FHIR Object Creator
- fhirInterface: FhirInterface
+ addActionToPlan(planID, action): void
+ addAllActionToPlan(planID, actionList): void
+ createPlanDefinition(PlanDefinition): String
FHIR Store
(from Package Structure)
Workflow Engine
(from Package Structure)
Knowledge Model Controller
+ addWorkflowXML(file): File
+ createFHIRObjects(File): void
+ deployBpmnDiagram(XMLFile): String
+ deployDmnDiagram(XMLFile): void
+ deployModel(XMLFile, bpmnName): void
Workflow Element Extractor
- doc: Document
+ getDataAssociations(): List<String>
+ getDiagramInfo(): PlanDefinition
+ getReceiveTasks(): List<Action>
+ getUserTasks(): List<Action>
XML Workflow manipulator
- doc: Document
- fileName: String
+ addXMLTag(): void
+ getElement(): void
+ saveBPMNModel(): File
Characterizing Classes
Information Holder
Interfacer
Service Provider
Controller
Structurer

Figure 5.3: Knowledge Model Manager - Class Diagram

Workflow Capability Core

The Workflow Capability Core controls knowledge models and synchronizes the Execution Module with the Data Module by updating them with the appropriate Workflow state. Additionally, the Workflow Capability Core contains functionality to support the off-the-shelf Workflow Engine used in the Execution Module. The full Workflow Capability Core is presented in Figure 5.4. The main component of the Workflow Capability Core is the SubscriptionController, which exposes an interface for the FHIR Store and the Workflow Engine to connect to. The subscription controller manages requests and interacts with the appropriate API. E.g., the function *finishWorkflow()* is a request from the Workflow Engine to the Workflow Capability Core to clean up the Workflow Instance in the FHIR Store, hence the Subscription Controller sends a status change request to "completed" to the FHIR Store for the appropriate CarePlan instance. Lastly, we see a Factory pattern to instantiate a ConcreteEngineInterface. This ConcreteEngineInterface depends on the Workflow Engine used and implements all the functionality the Workflow Capability expects from a workflow engine.

Design Decision: According to requirement **NF-03** and **NF-03a**, the system should have a modular design which prevents a Workflow Engine vendor lock-in. To accommodate this, instead of directly interacting with the Workflow Engine API, we chose to go with a Factory Pattern. In this way, we define our expected API functionality in the AbstractEngineInterface and point towards the right Workflow Engine API calls in the ConcreteEngineInterface. In this way, we don't have to rebuild our Workflow Capability Core when adding or exchanging a Workflow Engine.

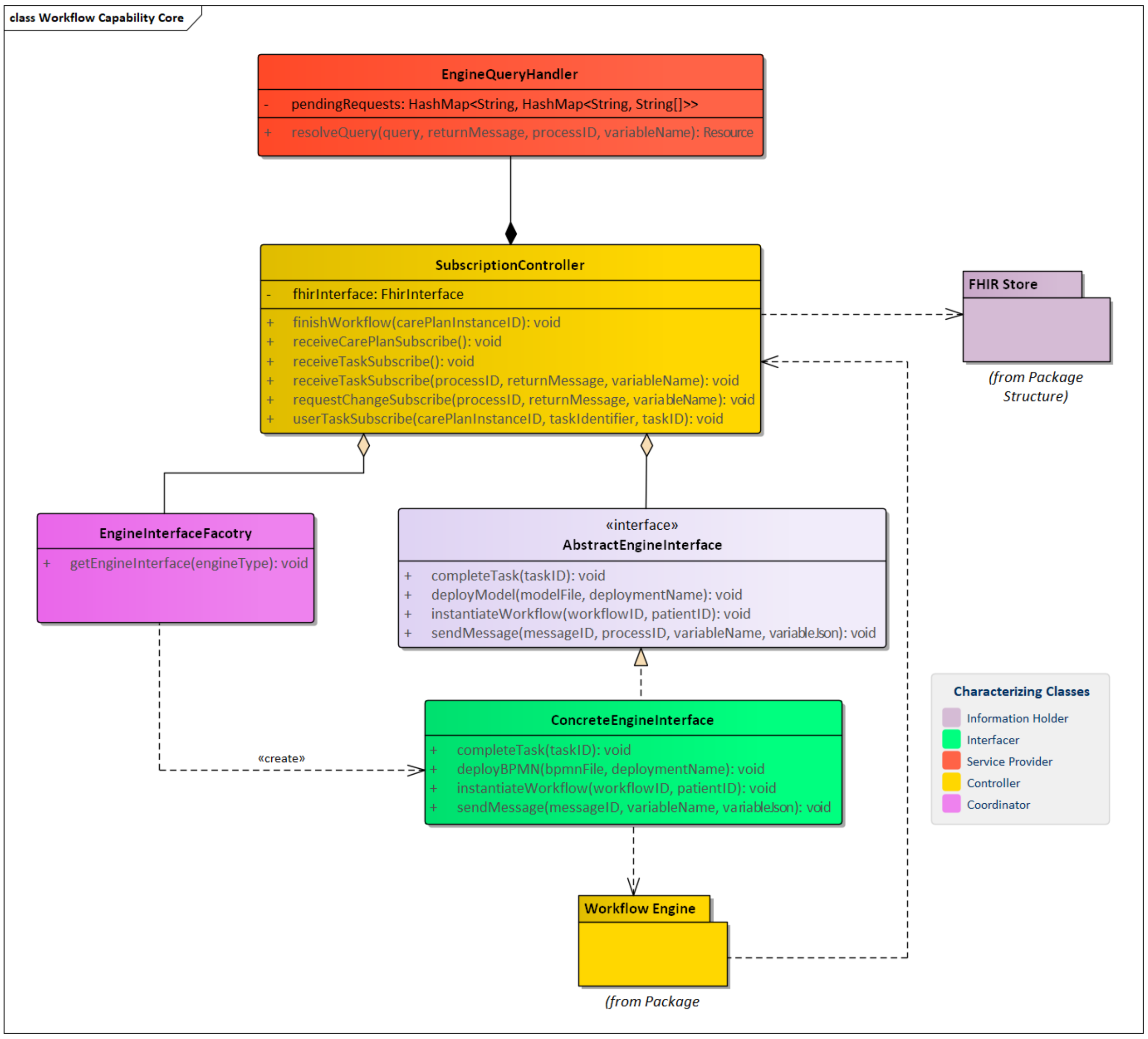

class Workflow Capability Core
EngineQueryHandler
- pendingRequests: HashMap<String, HashMap<String, String[]>>
+ resolveQuery(query, returnMessage, processID, variableName): Resource
SubscriptionController
- fhirInterface: FhirInterface
+ finishWorkflow(carePlanInstanceID): void
+ receiveCarePlanSubscribe(): void
+ receiveTaskSubscribe(): void
+ receiveTaskSubscribe(processID, returnMessage, variableName): void
+ requestChangeSubscribe(processID, returnMessage, variableName): void
+ userTaskSubscribe(carePlanInstanceID, taskIdentifier, taskID): void
FHIR Store
(from Package Structure)
EngineInterfaceFacotry
+ getEngineInterface(engineType): void
«interface»
AbstractEngineInterface
+ completeTask(taskID): void
+ deployModel(modelFile, deploymentName): void
+ instantiateWorkflow(workflowID, patientID): void
+ sendMessage(messageID, processID, variableName, variableJson): void
«create»
ConcreteEngineInterface
+ completeTask(taskID): void
+ deployBPMN(bpmnFile, deploymentName): void
+ instantiateWorkflow(workflowID, patientID): void
+ sendMessage(messageID, variableName, variableJson): void
Workflow Engine
(from Package
Characterizing Classes
Information Holder
Interfacer
Service Provider
Controller
Coordinator

Figure 5.4: Workflow Capability Core - Class Diagram

5.4.4 Execution Module

The Execution Module has the purpose of running the executable elements of a Knowledge Model. As a result of the exploration described in Section 2.3.1, we choose the BPM+ Health methodology of combining the BPMN, CMMN, and DMN standards to express Knowledge Models. This means that the executable workflow elements can consist of multiple of three different entities: BPMN models, DMN models, and CMMN models. To make use of these entities, we would need a software product to interpret and execute these models. With such a large product, we have to consider the make-or-buy decision. In prior exploration [2], Philips has decided to go with an off-the-shelf solution, because manufacturing a Workflow Engine for this purpose would be too costly. However, as we have seen in Figure 2.1, the maturity and availability of different tools for the BPM+ Health standard are quite low. We believe, a mature off-the-shelf Workflow Engine that combines all three model types doesn't exist. Nevertheless, a combined BPMN and DMN engine does exist, as well as an off-the-shelf CMMN engine. The modular design of the Control Model allows for multiple Workflow Engines to be accessed by creating multiple ConcreteEngineInterfaces pointing to different engines. This makes running more than one engine accessible.
The current prototype has a focus on BPMN and DMN for which the off-the-shelf Camunda engine is fitting. The implementation of combining multiple engines is out of scope for the current prototype.

5.5 Process view

For the Process view, there are several dynamic aspects to consider. We will discuss how the interaction between the modules in Figure 5.2 operates for certain processes. Section 5.5.1 discusses the process of deploying a workflow model and how the workflow data itself is processed. In Section 5.5.2 we discuss what the process of interacting with a Workflow through the Application Module looks like. Finally, Section 5.5.3 explains the process of the usage of data in workflows.

5.5.1 Model Deployment

For the Model Deployment process, a Workflow Producer uploads the Knowledge Model specification to the Deployment Application. In the current prototype, we allow the deployment of XML definitions of BPMN and DMN models. This definition is sent to the Knowledge Model Manager which concretely does two things:

- Manipulate the XML definition, e.g. by adding certain tags, such that it's interpretable by the Workflow Engine used.
- Map the XML model to FHIR Objects, as specified in the FHIR Mapping (Section 5.3.

After this process, the output data is used by the Workflow Capability Core to delegate it to the Workflow Engine and the FHIR Store. When the model definition data is available in both places, the model is successfully deployed. Direct interaction of the modules for the mapping and creating the FHIR Objects, is presented in the Sequence Diagram in Figure 5.5.

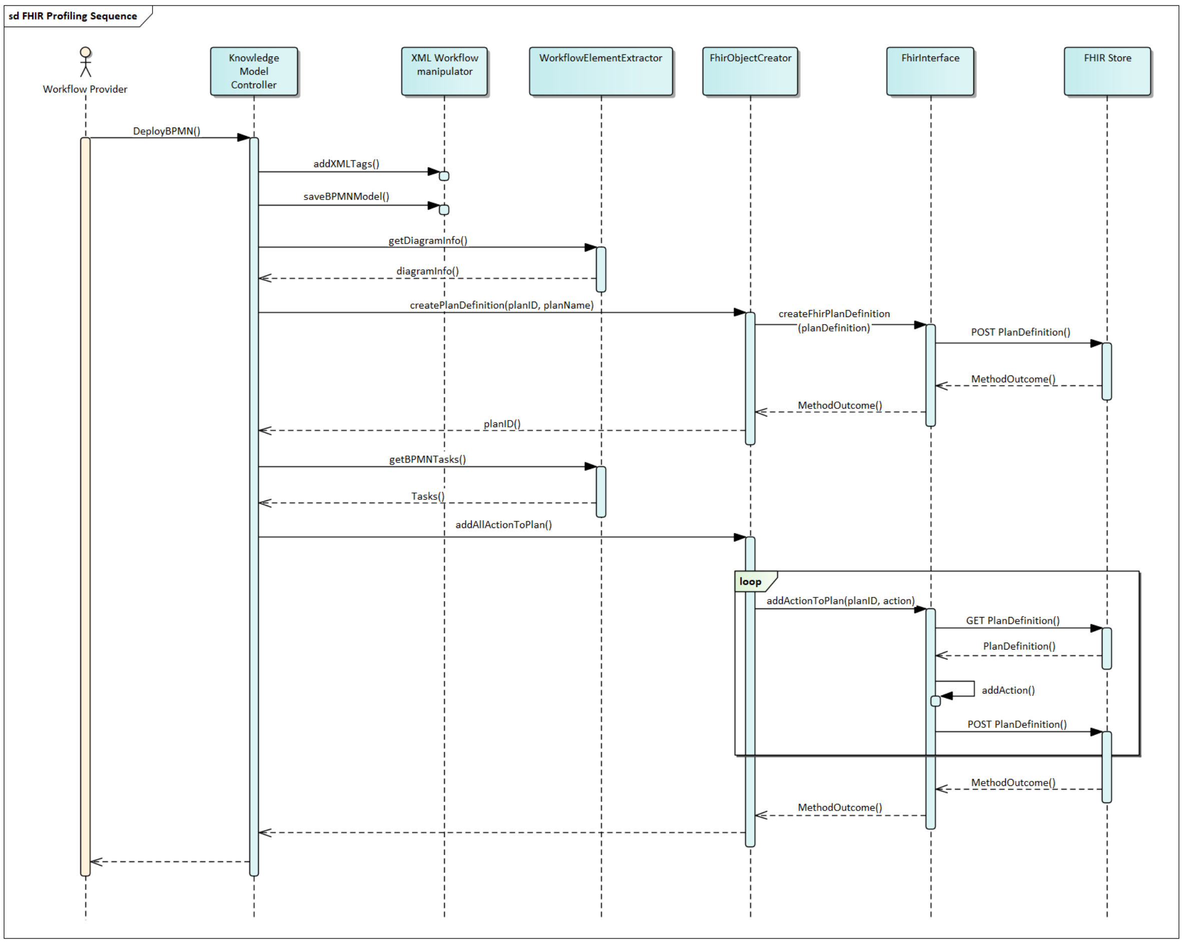


Figure 5.5: Model Deployment FHIR Mapping - Sequence Diagram

5.5.2 Workflow Interaction

One of the main advantages of an executable workflow is that it's dynamic based on Workflow Interaction. This interaction is done by the Caregiver. In the scope of our system, there are two main processes involved with regards to Workflow Interaction:

Workflow Initiation

Workflow Initiation is the process of starting an instance of a workflow definition for a certain patient. The Workflow Initiation process is presented in a Sequence Diagram in Figure 5.6. This process starts in the Guidance Application where the Caregiver selects a Workflow Definition that needs to be instantiated for a certain subject, which is the Patient. The application then signals an "Apply" request to the FHIR Store. This Apply request is predefined behavior of the FHIR Store, specified in the PlanDefinition FHIR Specification [27]. The apply operation depicts the elements of a PlanDefinition to a new CarePlan Resource. This mapping between definition and instance was earlier discussed in Table 5.1, where also the CarePlan structure is presented. On every new creation, the FHIR Store signals the Workflow Capability Core that a new CarePlan has been created. On receiving this signal the Workflow Capability Core starts the workflow in the Workflow Engine by interacting with its API.

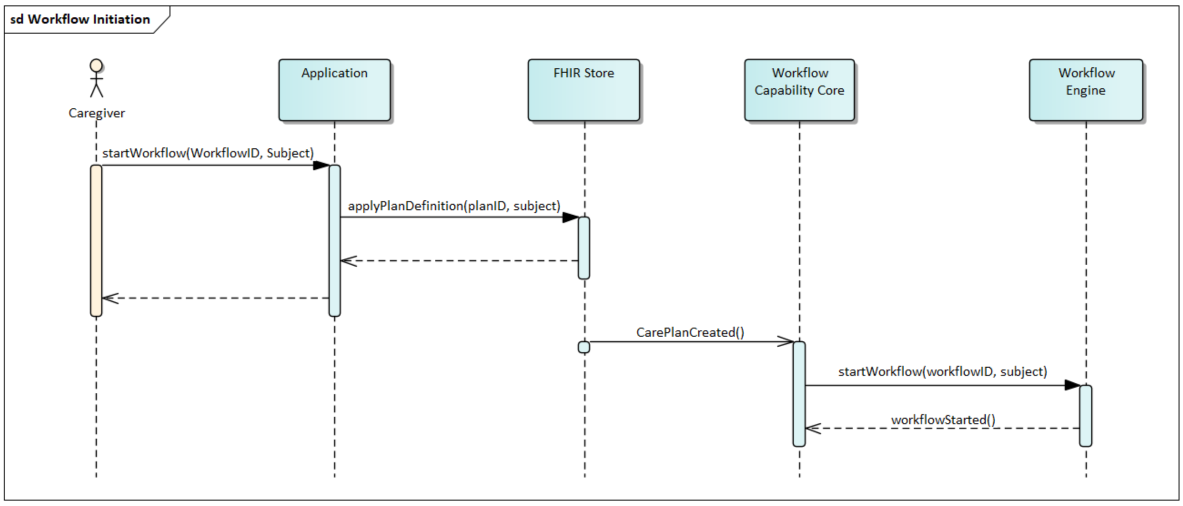


Figure 5.6: Workflow Initiation - Sequence Diagram

Task Interaction

Task Interaction is the process of giving input to and getting output from running Tasks in the workflow instance. The first important task is to make information about the state of a Workflow available in the FHIR Store, since that's the only exposed data interface to the Application Module. This is done by marking a Task FHIR Resource with a status, either "ready" or "requested". The process for this is as follows, The Workflow Engine signals the Workflow Capability Core that a Task is entered and therefore requested, after which the Workflow Capability Core updates the status value of the Task FHIR Resource in the FHIR Store. The full process is presented in Figure 5.7.

After this step the data about the task is available in the FHIR Store and can be fetched by the application. Next, the Caregiver should be able to signal that a Task is completed, which is done through the Clinical Application. Again, the status of a task is changed in the FHIR Store, this time to "completed". The FHIR Store signals the HealthSuite Workflow Core about the change in Task, after which the HealthSuite Workflow Core signals the Workflow Engine that the task is finished.

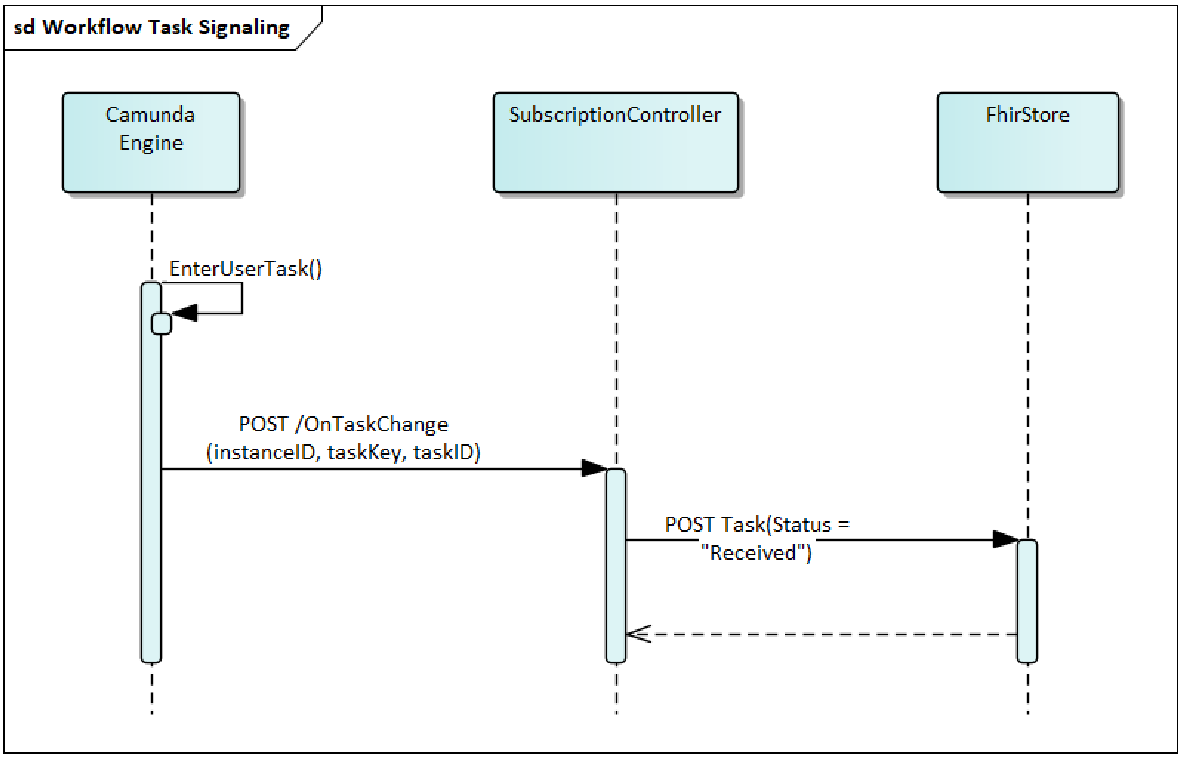

sd Workflow Task Signaling
Camunda Engine
SubscriptionController
FhirStore
EnterUserTask()
POST /OnTaskChange (instanceID, taskKey, taskID)
POST Task(Status = "Received")

Figure 5.7: Workflow Task Signaling - Sequence Diagram

Design Decision: For the instantiation of tasks we can make the choice between instantiating all tasks together with the instantiation of the CarePlan or instantiating every task separately upon entering the task in the process. Initially, we went with the first approach, since it seemed beneficial to see the status of upcoming tasks before entering them. However, after some experiments we found that this brings several problems:

- **Multiple token Issue:** When defining Tasks beforehand, a one-to-one mapping is made from the actions in the *PlanDefinition* to Tasks. This means for every action exactly one Task is created. In BPMN multiple tokens can enter a single task, which means two instances of the same task can run at the same time, possibly with a distinct status.
- **Information excess:** When defining Tasks beforehand, all tasks in the definition are initialized. This means that even Tasks that never will be executed will be initialized. Preferably, when fetching all initialized Tasks, you would want Tasks that are being executed or that have been executed in the past.

5.5.3 Usage of Clinical Data in Workflows

This section describes the process of a workflow running in the workflow engine using the clinical data available in the FHIR store. This process is presented in a sequence diagram in Figure 5.8.
When the Workflow Engine enters a ReceiveTask with an attached Data Object, it means the workflow needs external data. The workflow Engine sends a signal to the Subscription controller with the query of the data needed and a return message ID. The Subscription then resolves this query and fetches the

requested Resource from the FHIR Store. In the case of a GET operation in the query, the FHIR Store immediately returns the requested Resource. In the case of a SUBSCRIBE event, the FHIR Store returns the resource when an update to the Resource has occurred. Finally, after the Subscription Controller received the Resource it sends the data as a message to the Workflow Engine.

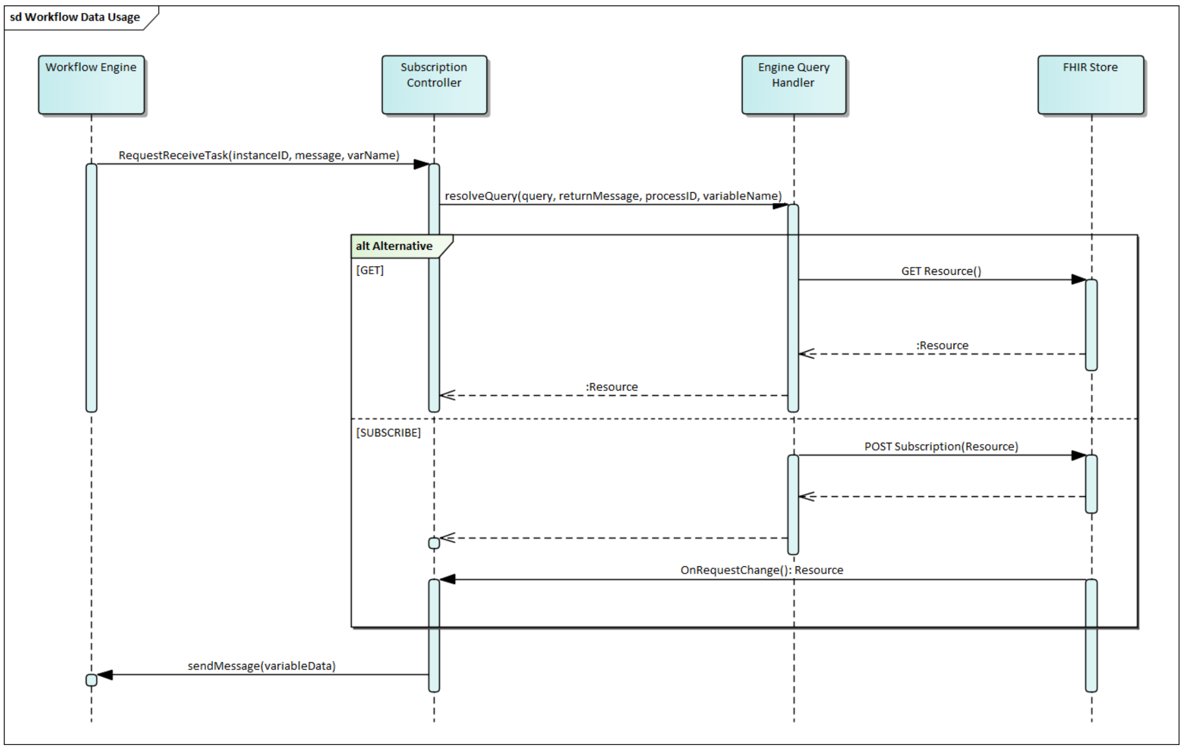

sd Workflow Data Usage
Workflow Engine
Subscription Controller
Engine Query Handler
FHIR Store
RequestReceiveTask(instanceID, message, varName)
resolveQuery(query, returnMessage, processID, variableName)
alt Alternative
[GET]
GET Resource()
:Resource
:Resource
[SUBSCRIBE]
POST Subscription(Resource)
OnRequestChange(): Resource
sendMessage(variableData)

Figure 5.8: Workflow Data Usage - Sequence Diagram

5.6 Physical view

The Physical view presents an overview of the software distribution and deployment of the system. In the current version of the prototype, all components are locally deployed as separate services. These services communicate through HTTP using the REST protocol. A deployment diagram of the current prototype implementation is presented in Figure 5.9.

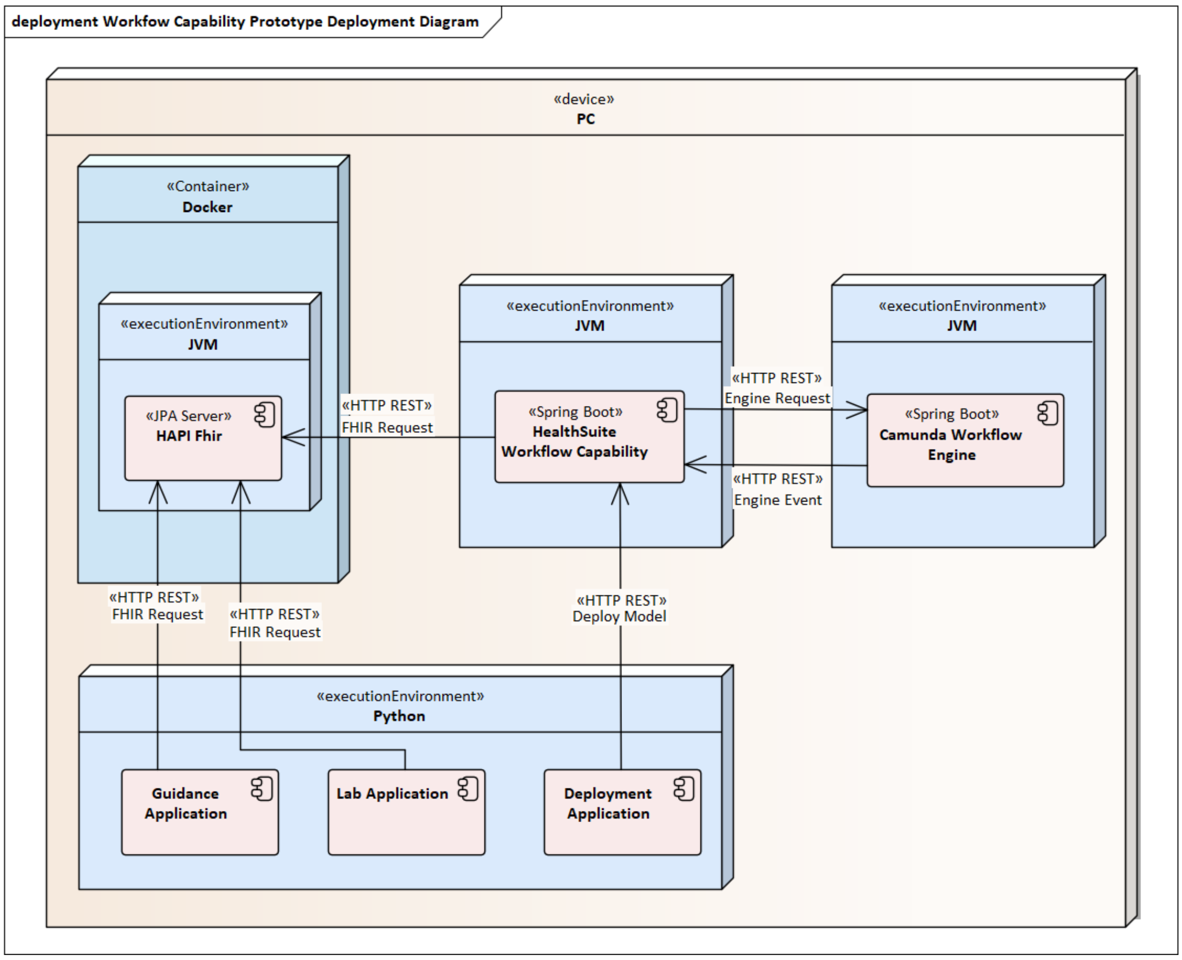

deployment Workfow Capability Prototype Deployment Diagram
«device» PC
«Container» Docker
«executionEnvironment» JVM
«JPA Server» HAPI Fhir
«HTTP REST» FHIR Request
«executionEnvironment» JVM
«Spring Boot» HealthSuite Workflow Capability
«HTTP REST» Engine Request
«HTTP REST» Engine Event
«executionEnvironment» JVM
«Spring Boot» Camunda Workflow Engine
«HTTP REST» FHIR Request
«HTTP REST» FHIR Request
«HTTP REST» Deploy Model
«executionEnvironment» Python
Guidance Application
Lab Application
Deployment Application

Figure 5.9: Workflow Capability Prototype - Deployment Diagram

Design Decision: The reason why REST is chosen as a communication protocol is twofold. Firstly, requirement **NF-04** states that the system must be easy to integrate by using standardized communication protocols. REST is a well-known and simple-to-understand communication protocol that satisfies this requirement. Secondly, using REST APIs makes the design modular in the sense that components can be easily interchanged if they provide similar interfaces expected by other components. For the Workflow Engine, this is required because of requirement **NF-03**. However, this is also convenient for other components, such as the FHIR Store. Implementations of HL7 FHIR Databases are still in early development, which could mean interchanging such a component can be beneficial in the long run.

As mentioned before, the Workflow Capability solution is envisioned to be a part of the HealthSuite Digital Platform (HSDP). The HealthSuite Digital Platform is a unique cloud-based platform with a secure, public API. HSDP has a cloud-first approach, hence if we want to integrate our system in HSDP we must allow for cloud-based deployment. This has been taken into account in the design of the system by deploying the different system components as different services, communicating through a standardized HTTP protocol. An example of a future envisioned deployment of the Health-Suite Workflow Capability is presented in Figure 5.10. Take note that other HealthSuite systems have not been considered in this envisioned solution. In the envisioned deployment we make use of Amazon Web Services (AWS), which is the standard cloud service Philips uses for all of its cloud products. One of the main differences from our prototyped deployment is the replacement of HAPI FHIR with the native FHIR Works service offered by Amazon. Another difference is the interaction with the FHIR Store which is now done through the Amazon API Gateway. Nothing drastically changes with regard to the system architecture. However, the implementation must make sure that all calls to the FHIR Store are accepted by the Amazon API Gateway.

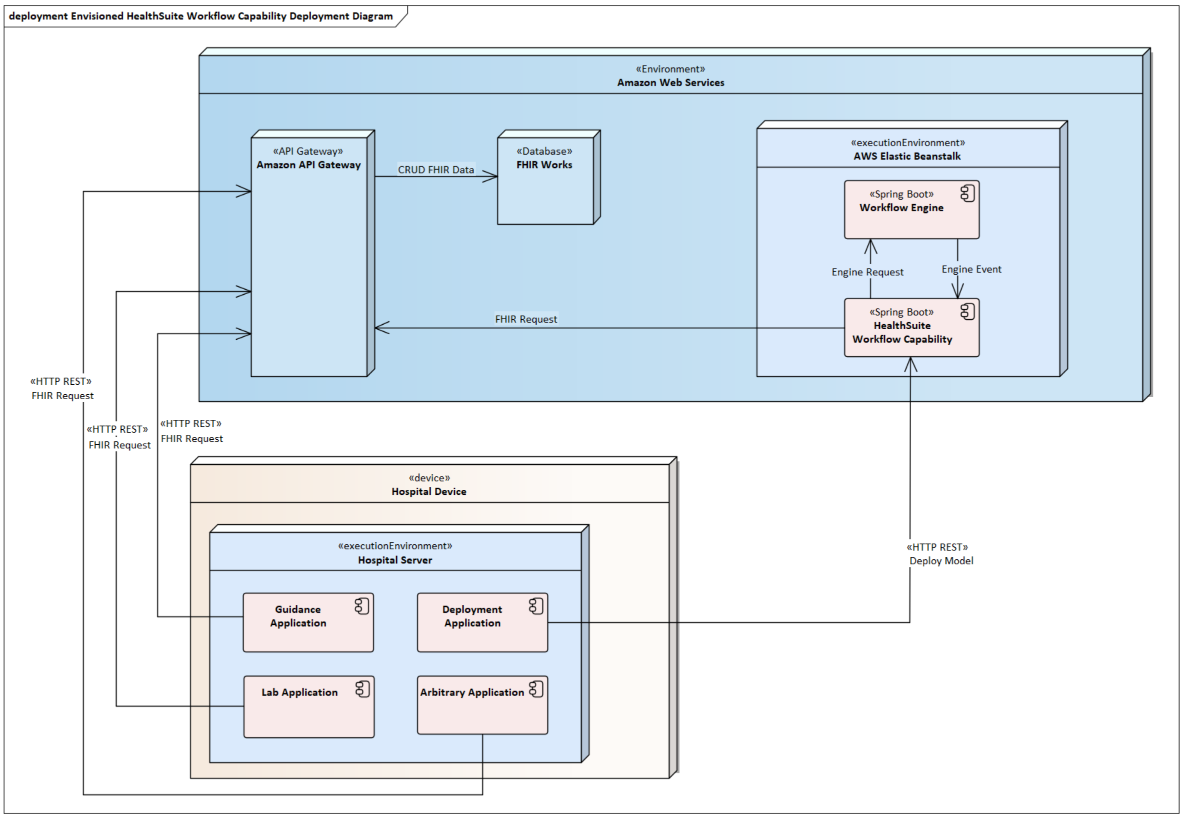

deployment Envisioned HealthSuite Workflow Capability Deployment Diagram
«Environment» Amazon Web Services
«API Gateway» Amazon API Gateway
CRUD FHIR Data
«Database» FHIR Works
«executionEnvironment» AWS Elastic Beanstalk
«Spring Boot» Workflow Engine
Engine Request
Engine Event
«Spring Boot» HealthSuite Workflow Capability
FHIR Request
«HTTP REST» FHIR Request
«HTTP REST» FHIR Request
«HTTP REST» FHIR Request
«device» Hospital Device
«executionEnvironment» Hospital Server
Guidance Application
Deployment Application
Lab Application
Arbitrary Application
«HTTP REST» Deploy Model

Figure 5.10: HealthSuite Workflow Capability - Envisioned Deployment Diagram

5.7 Development view

The Development View presents the static organization of the software. This section also describes how we make the system shareable in a software community to comply with requirement **NF-05**. The non-technical aspects of the development view, e.g. planning and way of working, are described in the Project Management section in Appendix A.
The software is shared as a repository in the Philips Innersource Github Organization, to make it accessible for all Philips employees. We will describe how the code structure follows from the design described in Section 5.4. The Github repository is split up into three folders, app, wfc, and engine. The content of these folders is described below.

The app folder

This folder represents the Application Module containing the Deployment Application, Clinical Application, and Lab Application. Though these applications can easily be separated, the choice was made to group them into a single web application to make the prototype more demonstrable and easy to use. The web application is written in python while using the Flask package. The app folder structure is presented in Figure 5.11.

```
app
├ cdr
│ ├ cdr_fhir.py
│ ├ dbconfig.py
│ └ wfci.py
├ web_application
│ ├ templates
│ │ ├ home.html
│ │ ├ patient.html
│ │ └ ......html
│ └ views.py
├ main.py
├ requirements.txt
└ service_config.py
```

Figure 5.11: The app folder tree structure

```
wfc/src/main/java/com/philips/workflowcapability
├ fhirresources
│ ├ FhirInterface.java
│ └ FhirPreProcessing.java
├ knowledgemodelmanager
│ ├ camundaInterface
│ │ └ CamundaXMLModifier.java
│ ├ BPMNElementExtractor.java
│ ├ FhirObjectCreator.java
│ └ KnowledgeModelController.java
├ wfccore
│ ├ CamundaInterface.java
│ ├ EngineInterface.java
│ ├ EngineInterfaceFactory.java
│ ├ EngineQueryHandler.java
│ └ SubscriptionController.java
└ WfcServiceApplication.java
```

Figure 5.12: The wfc folder tree structure

The cdr folder contains functionality to manage interaction with the "Clinical Data Repository", in our case the FHIR Store. The service_config.py file in the app folder contains the configuration to set the correct FHIR endpoint. The web_application folder contains the core functionality of the app. The views.py file contains the functionality for each routing made to the web application. The templates folder contains the UI components used by views.py.

The wfc folder

The wfc folder represents the Control Module, containing the Knowledge Model Manager and the Workflow Capability Core. It contains a Java Spring boot application. The file structure in the src folder is presented in Figure 5.12.

Firstly, the knowledgemodelmanager folder represents the Knowledge Model Manager package in the package diagram. The mapping of the files inside this folder is quite trivial since it's close to the class structure presented in Figure 5.3. The exact mapping of the classes in the diagram to the files is presented in Table 5.3.

Table 5.3: Knowledge Model Manager Mapping

Class from Class Diagram	**File from File structure**
Knowledge Model Controller	KnowledgeModelController.java
FHIR Object Creator	FhirObjectCreator.java
Workflow Element Extractor	BPMNElementExtractor.java
XML Workflow Manipulator	camundaInterface/CamundaXMLModifier.java

Secondly, the wfccore folder represents the Workflow Capability Core package in the package diagram. This means the files in this folder map to the Class Diagram presented in Figure 5.4. The exact mapping of the classes in the diagram to the files is presented in Table 5.4.

Table 5.4: Workflow Capability Core Mapping

Class from Class Diagram	**File from File structure**
SubscriptionController	SubscriptionController.java
EngineQueryHandler	EngineQueryHandler.java
AbstractEngineInterface	EngineInterface.java
EngineInterfaceFactory	EngineInterfaceFactory.java
ConcreteEngineInterface	CamundaInterface.java

Finally, the fhirresources folder is used to interact with the FHIR Store, the FhirInterface.java file is used by other files to make FHIR API calls. The FhirPreProcessing.java file fills up the FHIR Store with mock data. This file is not directly part of the system, however is used for convenience of demonstrating the prototype.

The engine folder

The engine folder represents the Execution Module and can contain one or more workflow engines to be used by the other modules. The Workflow Engine used in the current prototype is the Camunda Engine, for which we use a Spring Boot application [28]. The file structure in the engine folder is presented in Figure 5.13.

```
engine/camunda/src/main/java/org/camunda/bpm/
├-delegate
│ ├-FinishWorkflow.java
│ ├-ReceiveTaskEntry.java
│ ├-StartEventDelegate.java
│ └-UserTaskEntry.java
└-WebApplication.java
```

Figure 5.13: The engine folder tree structure

The WebApplication.java file is used to start the Spring Boot instance of the Camunda Engine. The files in the delegate folder are used to message the Workflow Capability Core package about certain events that occur. E.g. the implementation in UserTaskEntry.java signals the Workflow Capability Core package when a User Task is entered.

5.8 FHIR Query

Clinical workflows often need information about the Patient to make certain decisions on the route the workflow should take. This information is stored in the local hospital's FHIR Store. If we want to use the data within the Workflow Engine executable we need to find a way to query this data from the FHIR Store. According to the BPMN definition [11], data in BPMN models are represented with a *Data Object*. A Data Object can represent documents used in a process, either physical or digital. These Data Objects can be connected to another BPMN element using a *Data Association* arrow. Although Data Objects can be connected to any BPMN element using a Data Association, we chose to always connect a Data Object to a Receive Task. A full BPMN representation of a Data Object attached to a Receive Task is presented in figure 5.14.

A Receive Task in BPMN waits for a certain message before it passes on the token to the next element. This is a good fit for our use case since we don't want to continue the workflow before the queried data is available. When we enter a Receive Task with a Data Object attached, we send out a query to the Workflow Capability which handles the communication with the FHIR Store. When the data is retrieved a REST message is sent to the Workflow Engine, with the piece of data attached. When this message is received by the Receive Task, the data is stored locally in the workflow engine and can be used and referenced from that point on in the remainder of the model.
To query a FHIR Object, we must find a way to create a reference to the right FHIR Object, and find a location to store this reference in the BPMN model. A Data Object in the BPMN definition has few fields to put additional data in, in the base BPMN definition without any extensions it merely has a *Id*,

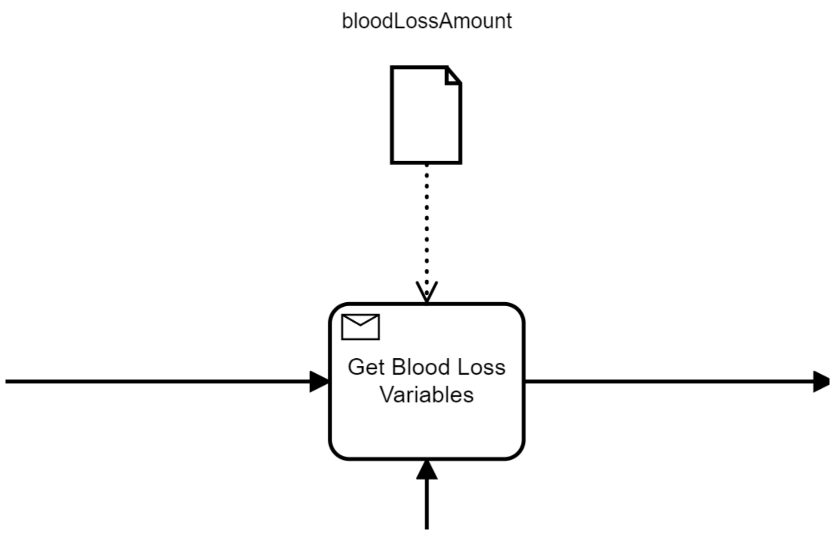


Figure 5.14: Receive Task with a Data Object attached through a Data Association

Name, and a *Documentation* field. This makes the Documentation field the most suitable for a freely interpretable query.
To reference the right FHIR Object, we chose a standardized FHIR Query format presented in figure 5.15. This format consists of five subparts: Data Type, CRUD Operation, FHIR Object, FHIR Object query, and a FHIR Path. Using this full standardized query format demands in-depth knowledge about the FHIR standard, which makes creating a standardized workflow tough. Therefore a trade-off can be made on whether some sub-parts are assumed to be a certain value, or they are omitted and assumed by the Workflow Capability. Below we explain the purpose of these sub-parts and the effects of omitting a sub-part.

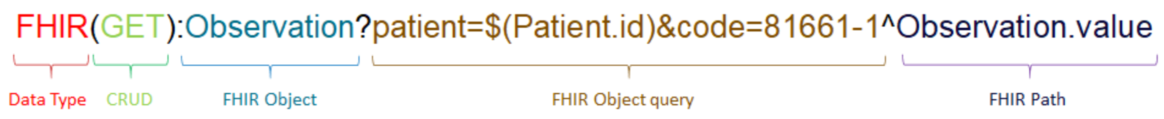

FHIR(GET):Observation?patient=$(Patient.id)&code=81661-1^Observation.value
Data Type
CRUD
FHIR Object
FHIR Object query
FHIR Path

Figure 5.15: Example of a detailed FHIR Query standardization

Data Type

The Data Type sub-part defines the type of object we will query. In the current prototype, we only accept FHIR data types, thus it can be said that having a Data Type sub-part in the current prototype is redundant. However, if future iterations of the HealthSuite Workflow Capability do accept different data types, this is an important field to have. If this field is omitted, we have to assume a certain data type, which currently makes sense, since we only use FHIR data.

CRUD Operation

Depending on the availability of data, different CRUD Operations can be used. In some systems, we directly want the current value of a FHIR Object, in which case we use a GET operation. In other cases, we want to wait for an update of a FHIR Object, in which case we use a SUBSCRIBE operation. If we want to extend the functionality of the FHIR Query to not only read values from the FHIR Store but also write to the FHIR store, we can use this field to specify a POST or PUT operation to the FHIR Store.
Omitting this sub-field means we need to assume the type of CRUD Operation used, which in the case of retrieving data would be a GET operation.

FHIR Object

To get the correct piece of FHIR Data, we need to specify which kind of FHIR Object we want to query. Almost always, a FHIR Observation needs to be queried for this. Observations are measurements and simple assertions made about a patient, device, or other subjects. There can be a few exceptions where we want specific data on the patient itself e.g., the Patients birth data, which would be stored in the FHIR Patient Object.
Omitting this sub-field means we assume the FHIR Object which needs to be queried, which logically here would be a FHIR Observation. However, this limits the use of this sub-field, since other Objects can't be referenced anymore. If this field is omitted, it should be made clear that only data stored in a FHIR Observation can be queried by a workflow.

FHIR Object query

Referencing the type of object isn't enough, we need to query one specific FHIR Object and not a FHIR Object type. For this, we need the FHIR Object query sub-part, which specifies which specific FHIR Object we want to be returned by defining certain parameters. This sub-part can use variables from the workflow engine local data by specifying it with the $ sign. In the example in Figure 5.15, we take the patient ID from the local engine data. In the current prototype, if the FHIR Object query points to multiple Objects, the most recent one is taken.
The FHIR Object query can never be fully omitted since this would mean we never point to a specific Object. However, we can make certain assumptions about the FHIR Object query e.g., we always want a FHIR Object with the reference to the current patient. In this way we don't have to specify the patient in the FHIR Object query, which makes it more usable, but less flexible. Additionally to the previous assumption, we could assume that we always want to specify a LOINC code of a current patient observation, in this way we never have to specify the code system we use.

FHIR Path

The FHIR Path further narrows the queried Object into a single value. So far we have queried a full FHIR Object, which is a JSON piece with many name/value pairs. By using the FHIR Path we can point to a specific name/value pair which is needed rather than querying the full FHIR Object.// The FHIR Path can easily be omitted, which would mean we don't query a specific piece of data, but rather query the whole FHIR Object. This means that we send a full JSON Object to the workflow engine, giving us an extra requirement for the workflow engine which we use, which is that they know how

to interpret JSON objects. Additionally while using the FHIR Object in the workflow, we still need to specify which name/value pair we need to evaluate something on.
If we assume that in the FHIR Object part of the query only Observations are queried, we could assume that we always need the **value** name/value pair from the Observation Object.

In the current prototype, we made certain decisions on the abstraction of these sub-parts, based on the current scope of the use case. We only support FHIR Data Types but did not omit the data type sub-part, because it indicates we want to query something from the FHIR store. Any other arbitrary documentation not starting with the FHIR Data type will be ignored in the query resolver. For the CRUD operations, we chose to go with the GET and SUB operations since for this prototype we want to support getting immediate data and waiting for a data change to occur. Nothing is omitted from the FHIR Object or FHIR Object query, given that this is a prototype we don't want to limit these queries yet, usability can be improved with assumptions in a later stage of the HealthSuite Workflow Capability development. Lastly, the FHIR Path is omitted from the FHIR Query. The workflow engine used supports JSON values, this makes it more convenient to query a whole FHIR Object and refine in the model itself.

6 Verification and Validation

In this chapter, we discuss how the Workflow Capability Prototype is verified and validated. Section 6.1 describes the verification steps taken, it will discuss Unit Tests, Integration Tests, and Test Automation. Section 6.2 describes the validation steps taken, in particular, we discuss the process of stakeholder feedback and the Workflow Engine Modularity Validation step.

6.1 Verification

According to the CMMI for Software Engineering [29], verification is defined as such: *Verification confirms that work products properly reflect the requirements specified for them. In other words, verification ensures that "you built it right."*. This means that we continuously have to check whether what we build adheres to the specifications set. To achieve this, we've performed Unit Tests and Integration Tests during the development of the system. To make sure all tests still hold at any stage of the development process a system for automated testing has been set up.

6.1.1 Unit Tests

Unit Tests is a testing method that verifies whether individual components have their expected behavior. Typically, these are small pieces of code that check the input and output of individual functions. In this way, when a test fails, the error can be easily traced down to the source.
The system built contains two off-the-shelf components, namely the Camunda engine and the Hapi FHIR implementation. While bringing in off-the-shelf components brings certain risks with respect to expected functionality, the examination of these products is out of scope for this project. However, both components are fully open-source, which means in the future they can be fully examined if needed.
Additionally to the off-the-shelf components, the system also contains multiple user applications, which mostly consist of UI elements. This leaves us with two main components that should be extensively Unit Tested, the Knowledge Model Manager and the Workflow Capability Core. These two components are implemented in Java and tested with Unit Tests executed by the Maven Surefire Plugin.

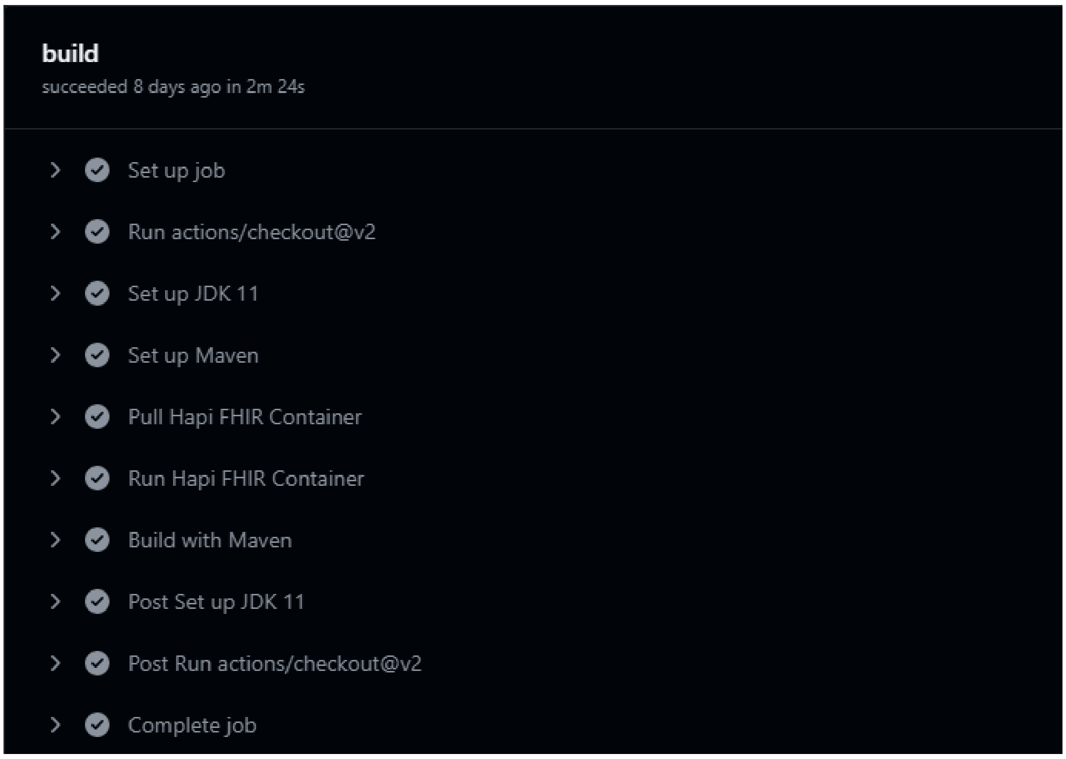
6.1.2 Integration Tests

When individual components are combined, Unit Testing often doesn't suffice. Unexpected behavior could occur because of interaction between components. Integration of components can be extra challenging when these components are developed by different persons and/or when components are

integrated at a late stage of development. To prevent the Integration risk in later stages of the project, the first deliverable was to get to a minimal end-to-end implementation of the HealthSuite Workflow Capability. By incrementing this minimal end-to-end implementation we ensured integration was done at an early stage, preventing the risk at a later stage in the project.
Additionally, where possible, we test interfaces between components, i.e. whether the creation of FHIR Objects is successfully saved as expected in the FHIR Store. Technical details on how this process was set up are further explained in Section 6.1.3.

6.1.3 Automated testing

During the development process of the system, the source code constantly changes. This means that when a test was once successful, it doesn't ensure it will be successful in the future. This means testing has to be repeated regularly. To make this process more effortless, at the beginning of the project a Continuous Integration (CI) pipeline has been set up in the Github environment. While using the Github Actions feature, we were able to set up a sequence of steps that simulates our running environment and where our tests can be run automatically. All steps taken by Github Actions are presented in Figure 6.1.


build
succeeded 8 days ago in 2m 24s

> ✓ Set up job
> ✓ Run actions/checkout@v2
> ✓ Set up JDK 11
> ✓ Set up Maven
> ✓ Pull Hapi FHIR Container
> ✓ Run Hapi FHIR Container
> ✓ Build with Maven
> ✓ Post Set up JDK 11
> ✓ Post Run actions/checkout@v2
> ✓ Complete job

Figure 6.1: Continuous Integration pipeline in Github Actions

The CI pipeline sets up the proper environment and pulls an empty instance of the Hapi FHIR implementation. In this way the interface between the Workflow Capability Core and the FHIR Store can be tested. After execution the full pipeline either succeeds or fails, which gives a direct indication whether something in the system broke down. The CI pipeline is ran on a self-hosted runner supplied by Philips. The CI pipeline was run on every push to the develop branch, or every pull request made, to make sure the tests were done regularly.

6.2 Validation

According to the CMMI for Software Engineering, validation is defined as such: *Validation confirms that the product, as provided, will fulfill its intended use. In other words, validation ensures that "you built the right thing."*. This means we must make sure the product developed is in line with the expectations of the stakeholders. Additionally, we must validate whether claims or assumptions made in the beginning still hold.

6.2.1 Stakeholder Feedback

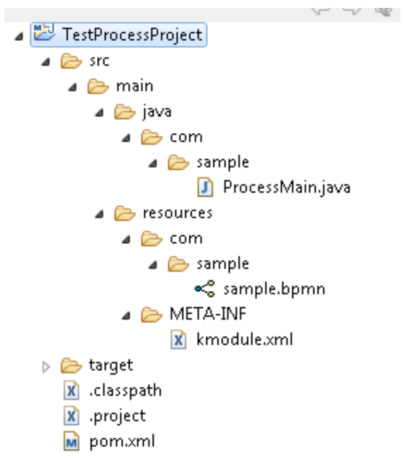
As described in the Project Management Appendix A, regular meetings were planned with the company supervisors, which represent all direct stakeholders. These meetings were used for stating progress, but also to align ideas about the system. Every time a considerable change to the system had been made, a demonstration of the system was given, which caused for a short feedback loop. Additionally, with every considerable change a pull request was made, which would be reviewed by one of the company supervisors. In this way, requirements such as **NF-05**, the shareability in a software community, were also validated. This process made sure the expectations of the stakeholders were as close to the developed product as possible.

6.2.2 Workflow Engine Modularity Validation

One of the most important aspects of the design presented is the claim that it has a modular design that prevents a vendor lock-in. The prototyped solution is verified and works with the Camunda workflow engine. However, **NF-03** states that the system should not have any hard dependencies on a Workflow Engine vendor. To support this claim, we must validate the modularity of the system by replacing the workflow engine vendor.
In this validation step, we chose to replace the Camunda engine with jBPM [15]. JBPM is a toolkit for building business applications, supporting BPMN2 and DMN models, similar to Camunda. The requirement we set on a workflow engine vendor is that it should support interfaces defined in the AbstractEngineInterface class (Workflow Capability Core - Class Diagram in Figure 5.4). Due to time constraints, we weren't fully able to deliver a prototyped solution containing jBPM as a Workflow Engine solution. However, we validated the jBPM interfaces and reported these findings back to Philips. One unexpected finding has to do with deploying the BPMN and DMN models to jBPM, which appears to be fundamentally different than deploying the same models to Camunda. This finding will be described in more detail in the next section.

JBPM Model Deployment Findings

While connecting the interfaces defined in the Workflow Capability Core, we encountered an unexpected difficulty. Where the Camunda engine expected raw XML files for the deployment of a model



```
TestProcessProject
  src
    main
      java
        com
          sample
            ProcessMain.java
      resources
        com
          sample
            sample.bpmn
        META-INF
          kmodule.xml
  target
  .classpath
  .project
  pom.xml
```

Figure 6.2: jBPM - KJAR Structure

that could be sent with a REST call, jBPM didn't support such input.
Red Hat, the developer of jBPM, decided to use an artifact packaging known as a "Knowledge Java archive", in short KJAR, to define their workflow input. The KJAR is an extension of a regular Java archive (JAR) file. These KJAR can contain multiple model definitions, including BPMN and DMN models. An example of the KJAR structure is presented in Figure 6.2.
JBPM supplies a "business-central" where KJAR's can be manually structured by importing raw XML models and deploying them to the jBPM engine manually. According to jBPM's documentation creating a KJAR can be done through several maven commands [30]. Unfortunately, time didn't allow for implementing this automation process, hence we can't validate this is easy to automate.

Modularity Validation Conclusion

Although we weren't able to fully replace the workflow engine vendor in the given time, we did validate the interfaces and came to interesting findings. In the light of future development of the HealthSuite Workflow Capability, the concept of a KJAR can be interesting. In this prototype, the knowledge model consisted of BPMN and DMN models. However, if we want to follow the BPM+ Health approach, CMMN models will be added to the knowledge models. When the knowledge model grows, it becomes important to "glue" these models together and define how they should interact. The KJAR does this by using a configuration file. We can copy this idea for the knowledge model and learn from the KJAR's structure.
In conclusion, if the automation steps of the KJAR mentioned above, which look promising, can be implemented, we can successfully exchange the Camunda workflow engine with the jBPM engine. However, if this automation is not possible, we can conclude that since model deployment isn't an automatic process, jBPM doesn't fit our requirements for a suitable replacement of Camunda.

7 Conclusion

This chapter focuses on the conclusion of the project. In section 7.1 we give a summary of the achieved results, referring back to the research questions defined at the start of the project. Recommendations for Philips and Future Work is discussed in Section 7.2.

7.1 Summary

The aim of the project was to create a detailed software design and prototype of the HealthSuite Workflow Capability. At the start of the project the following Research Questions have been defined.

RQ-01 Based on the high-level conceptual HealthSuite Workflow Capability Reference Architecture, what should a detailed software design for the HealthSuite Workflow Capability look like?
To answer this question, in this project, we have defined a detailed design in Chapter 5 providing the following value:

- A design with modularity of the key components in the system, ensuring that we have no hard dependencies on a Workflow Engine vendor.
- A design that allows for deployability in the cloud, which is essential for integrating the solution in the HealthSuite Digital Platform (HSDP).
- An interface definition between the Workflow Engine and the FHIR Store, which allows for Clinical Data to be used in the Workflow Engine executable.

To show the design is realizable, a prototype has been produced, implementing the detailed design.

RQ-02 How can the elements of the BPMN standard be mapped to the HL7 FHIR standard, such that software applications can interact with workflows through the FHIR Store?
During the project, we have obtained knowledge about the way BPMN models execute and the structure of the Workflow Module of the HL7 FHIR standard. As a result, we have proposed a FHIR Mapping in Section 5.3. The FHIR Mapping proposes to replicate some of the relevant elements in the BPMN standard, and express them in the Workflow Module of the HL7 FHIR standard.

The results of these research questions combined with the prototyped detailed design show promise for creating a HealthSuite Workflow Capability. The delivered design and prototype can serve as a baseline for further expansion and research by Philips. In section 7.2 we discuss our recommendations for the HealthSuite Workflow Capability and discuss Future Work.

7.2 Recommendations and Future Work

In this section we discuss possible directions for the continuation of the HealthSuite Workflow Capability exploration.

7.2.1 CMMN Extension

Knowledge Models as envisioned in Section 2.3 contain clinical knowledge expressed in the OMG standards BPMN, CMMN, and DMN. The project, however, had restricted resources. As a result, the CMMN standard was not implemented in the current HealthSuite Workflow Capability prototype. We suggest to extend the current prototype to accept the CMMN standard. To extend the current prototype we recommend the following approach:

- Firstly, create a FHIR Mapping for the CMMN standard, similarly as done for the BPMN standard, described in Section 5.3.
- Secondly, integrate a CMMN workflow engine next to the current Camunda Engine
- Lastly, extend the Control Module implementation to accept interactions with the Workflow Engine.

7.2.2 Exploration of the Clinical Quality Framework

The current prototype of the HealthSuite Workflow Capability shows the feasibility of such a system. The design proposed gives a baseline for future research, meaning the design can be further extended and improved. We suggest to look at existing standardization and tooling to improve the currently proposed design. An example of such standardization and tooling is the Clinical Quality Framework (CQF). "The Clinical Quality Framework is a joint effort by the Clinical Decision Support and Clinical Quality Information Work Groups to identify, develop, and harmonize standards that promote integration and reuse between Clinical Decision Support (CDS) and Clinical Quality Measurement (CQM)."[31] Efforts and tooling of this community are related to the clinical workflow subject.
Due to the lack of time, the CQF couldn't be extensively explored. However, below we suggest possible improvements to the proposed design using the CQF. These suggestions can be included in future work.

CDS Hooks

CDS Hooks[32] is a specification that uses a hook-based pattern to invoke decision support from within a workflow. Using CDS Hooks a client can trigger a service which returns a card containing important clinical information taken from the FHIR Store. CDS Hooks hasn't been researched during this project, however, we believe using CDS Hooks can make a Guidance Application more interpretable for users of the system, providing them with more specific information on a certain Task, or information about measurements. Therefore, we suggest to explore the possibility of using CDS Hooks to extend the FHIR Resources currently used, researching whether the usability of the system improves.

Clinical Quality Language

FHIR Query proposed shows the possibility of usage of FHIR Data in workflows. It can be beneficial for users to use standardized and properly defined query languages rather than using a Philips specific FHIR Query language. The Clinical Quality Framework proposes a high-level, domain-specific language used to enable sharing of clinical knowledge called the Clinical Quality Language (CQL). This language is a superset of the FHIR Path definition. Since the FHIR Path definition is used in the FHIR Query proposal in Section 5.15, we recommend exploring the CQL for replacing the proposed FHIR Query. During this exploration, we suggest to look at the following Research Questions:

- Does using the Clinical Quality Language have any functional advantages over using the proposed FHIR Query? E.g. can CQL query for a combination of FHIR Resources or a specific subset of a FHIR Resource?
- Does using the Clinical Quality Language bring any limitations compared to the proposed FHIR Query? E.g. not supporting subscribe events of FHIR Resources.
- Will using the Clinical Quality Language make creating Knowledge Models more effortless and/or comprehensible for Workflow Producers?

About the author

Juan van der Heijden received his Bachelor's degree in Software Science from Eindhoven University of Technology in 2017. After his graduation he started a Master program in Computer Science and Engineering on Eindhoven University of Technology which was finished in 2019. His master's thesis "Defining a conversion layer between SysML models and Unity", involved creating a method to transform model data to a new environment, while also creating a 3D-model prototype. During his bachelor and master programs he worked as a high school teacher in Informatics. After his master he decided to join the PDEng Software Technology Program. During the program, he was involved as an engineer, configuration manager, and project manager in projects for CERN, Pixelfarming Robotics, and ESA.

Bibliography

[1] Ricardo Quintano, Prescott Klassen, Javier Espina, Asim Muhammad, and Hongchao Nie. Dynamic workflow white paper. Technical report, Philips Research, 2020.

[2] Wouter Peters, Patrick Bonné, Bas Bergevoet, Erik Moll. Exploration note healthsuite workflow capability. Technical report, Philips Research, 2020.

[3] Philips. Philips HealthSuite Digital Platform. `https://www.usa.philips.com/healthcare/innovation/about-health-suite`. [Online; accessed 12-August-2021].

[4] Bloomberg Law. Insight: The healthcare industry's shift from fee-for-service to value-based reimbursement. 2018.

[5] World Health Organization. WHO Forum on Health Data Standardization and Interoperability. Technical report, December 2012.

[6] HL7. FHIR standard - Release 4. `https://www.hl7.org/fhir/`. [Online; accessed 21-September-2021].

[7] HL7. HL7 Version 2 Product Suite. `http://www.hl7.org/implement/standards/product_brief.cfm?product_id=185`. [Online; accessed 26-September-2021].

[8] HL7. HL7 Version 3 Product Suite. `https://www.hl7.org/implement/standards/product_brief.cfm?product_id=186`. [Online; accessed 26-September-2021].

[9] HL7. CDA® Release 2. `http://www.hl7.org/implement/standards/product_brief.cfm?product_id=7`. [Online; accessed 26-September-2021].

[10] BPM+ Health. About BPM+ Health. `https://www.bpm-plus.org/about-us.htm`. [Online; accessed 10-August-2021].

[11] Object Management Group. Business Process Model and Notation (BPMN). Technical report, January 2011. Version 2.0.

[12] Object Management Group. Case Management Model and Notation (CMMN). Technical report, December 2016. Version 1.1.

[13] Object Management Group. Decision Model and Notation. Technical report, January 2021. Version 1.3.

[14] OW2. Orchestra. `https://projects.ow2.org/view/orchestra/`. [Online; accessed 10-August-2021].

[15] Red Hat. jBPM. `https://www.jbpm.org/`. [Online; accessed 10-August-2021].

[16] Red Hat. Camunda. `https://camunda.com/`. [Online; accessed 10-August-2021].

[17] Philips Healthcare. List of Philips Healthcare Solutions. `https://www.philips.co.uk/healthcare/solutions`. [Online; accessed 12-August-2021].

[18] World Health Organization. WHO guidelines approved by the Guidelines Review Committee. `https://www.who.int/publications/who-guidelines`. [Online; accessed 12-August-2021].

[19] Elsevier. Elsevier ClinicalPath (formerly Via Oncology). `https://www.elsevier.com/solutions/clinicalpath`. [Online; accessed 12-August-2021].

[20] Wikipedia. Thieme Medical Publishers. `https://en.wikipedia.org/wiki/Thieme_Medical_Publishers`. [Online; accessed 12-August-2021].

[21] P.B. Kruchten. The 4+1 view model of architecture. *IEEE Software*, 12(6):42–50, 1995.

[22] Object Management Group. OMG® Unified Modeling Language® (OMG UML®). Technical report, December 2017. Version 2.5.1.

[23] Mikko Kontio, Production Manager, Softera. Architectural manifesto: Designingsoftware architectures, Part 5. Technical report, February 2005.

[24] R.J. Wirfs-Brock. Characterizing classes. *IEEE Software*, 23(2):9–11, 2006.

[25] HL7. FHIR Workflow Module. `https://www.hl7.org/fhir/workflow-module.html`. [Online; accessed 21-September-2021].

[26] HL7. FHIR Observation standard - Release 4. `https://www.hl7.org/fhir/observation.html`. [Online; accessed 21-September-2021].

[27] HL7. FHIR PlanDefinition standard - Release 4. `https://www.hl7.org/fhir/plandefinition.html`. [Online; accessed 23-September-2021].

[28] Camunda. Camunda Spring Boot - Getting Started. `https://docs.camunda.org/get-started/spring-boot/`. [Online; accessed 23-September-2021].

[29] CMMI Product Team. Mmi for software engineering, version 1.1, staged representation (cmmi-sw, v1.1, staged). Technical report, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 2002.

[30] Red Hat. Manually create business application? `https://docs.jbpm.org/7.22.0.Final/jbpm-docs/html_single/#_manually_create_business_application`. [Online; accessed 25-September-2021].

[31] HL7. Clinical Quality Framework. `https://confluence.hl7.org/display/CQIWC/Clinical+Quality+Framework`. [Online; accessed 26-September-2021].

[32] HL7. CDS Hooks. `https://cds-hooks.hl7.org/`. [Online; accessed 26-September-2021].

A Project management

This section covers the most important parts of the project management, as defined at the start of the project.

A.1 Project Deliverables

This section includes a list of deliverables that were needed to successfully finish the project. These deliverables include both software deliverables and document deliverables.

A.1.1 Software Deliverables

These software deliverables are implemented in the same software repository, and are connected in design. However, since they serve a different purpose, they are described as different deliverables. Both software deliverables will be integrated in the Philips Internal Codebase, therefore adhering to the Philips Internal Codebase code quality. A full list of software deliverables is presented in Table A.1

Table A.1: Software Deliverables

Deliverable ID	**Deliverable Description**
SD-01	Software demonstration of the integration method between the workflow definition and the FHIR Store.
SD-02	Software demonstration of a knowledge model expressed in process and decision models, executed in one or more engines.

A.1.2 Document Deliverables

The Document Deliverables serve as assistive material to the Software Deliverables or the project as a whole. The full list of Document Deliverables, as originially defined, is presented in A.2. Deliverables **DD-01** to **DD-04** are supportive documentation for Philips. Deliverable **DD-05** is supportive documentation for the TU/e

Table A.2: Document Deliverables

Delierable ID	Deliverable Description
DD-01	**Project Management Plan:** Describes how the project will be executed. Containing the project overview, context and planning.
DD-02	**Requirements Document:** Describes the needs for the product, both functional and non-functional, all prioritized.
DD-03	**Architecture Document:** Describes the system design, split in modules. If certain design decisions are made, they will be argued and compared to alternatives.
DD-04	**Production Document:** Describes the necessary steps to build, optimize, and deploy the built system. This includes the verification of the product (Test Cases).
DD-05	**Project Thesis:** Finalization document as a requirement by the TU/e. Describes everything done in the project. This document will overlap with the previously described documents.

A.2 Schedule and Budget Summary

The project date range is January 1st, 2021 until October 28th, 2021. The project uses an agile approach, meaning that the scope and deliverable adapts in content and size depending on the project progress and findings. There are several deadlines set by the TU/e, taken from the Final Projects Guide. The project context-related ones are listed here:

- March 1, 2021: Submit Project Management Plan
- March 31, 2021: Submit evaluation form for 'Initial Performance Evaluation'
- May 1, 2021: Submit 1st Draft of your Final Report to be reviewed by TEC members
- June 30, 2021: Submit evaluation form for 'Intermediate Performance Evaluation'
- September 20, 2021: Submit Excel form 'Data for Diploma and Ceremony 2021'
- 7 days before final presentation: Submit Final Technical Report

A.2.1 Features

Sprints contribute to larger features defined during the project. The list of features were not fully defined at the start of the project, but a list of features were actively managed by the PDEng trainee. Below all initial features with their acceptance criteria are listed.

Feature 1: Get to a minimal end-to-end implementation of the HealthSuite Workflow Capability

Description: To get a baseline for the research questions, a minimal end-to-end implementation of the HealthSuite Workflow Capability must be made. This minimal implementation is based on a specific

Use Case. For the minimal implementation, we choose the treatment of blood loss as a use case. The minimal implementation must consist of one or more models representing the use case, executable in a workflow engine, connected with FHIR.

Goals:

Delierable ID	Deliverable Description
F1G1	By implementing all described parts, get a basic understanding in all elements in the HealthSuite Workflow Capability. • Understand how to write knowledge models for a specific use case. • Understand how the described engines work. • Understand how the connection between workflows and FHIR is made.
F1G2	Have a software repository, containing a simple end-to-end implementation, where future features can build upon.

Deliverables:

Delierable ID	Deliverable Description
F1D1	**Design Document**: A document containing the findings and design decisions made for the minimal implementation
F1D2	**Software Demo:** Demonstrable implementation of the defined knowledge model in a workflow engine, with a clear connection to FHIR. The connection to FHIR can be shown by using a simple User Interface or showing the change of data in the FHIR store.

Acceptance Criteria:
F1D2 - Design Document

- Contains a complete UML description of the implementation.
- Design decisions are argumented, taking alternatives into account.
- Reviewed and accepted by the client.

F1D3 - Software Demo:

- Stored in GIT.
- Must have complete set of test cases, covering the requirements as much as possible.
- Reviewed and accepted by the client.

B HealthSuite Workflow Capability Requirements

Below all requirements defined in the Exploration Note HealthSuite Workflow capability[2] are listed.

Workflow orchestration requirement	**Description**	**MoSCoW**
methodology matches clinical practice	the workflow model needs to be understood by the users - in our case mostly clinical staff.	M
can support strictly procedural parts	like BPMN	S
can support unstructured, dynamic planning	like CMMN	S
can support decision tables	like DMN	S
has a graphical presentation format	a pure textual format (JSON, XML) will not be understood easily	M
customers can update the defined workflows	It should be easy to make small modifications for customization purposes.	S
data from various sources can be used in the workflow	workflow engine/tool should make that easy. Workflow model should include case/context data model.	M
workflows can be personalized for individual cases	e.g. SRC requirement for CareOrchestrator: customization of rules per patient is possible. Of course personalization can be made a part of the workflow as well...	S
tool should support proper development cycle (design, implement, test, release)		C
tool should support sending notifications (email, sms, mobile app notifications)		C
tool/engine can cope with high loads	SRC supports 9 million patients; workflow engine must be able to support high numbers of concurrent cases.	M
workflow definitions can be tuned per customer	question will be how this can be done in a maintainable fashion.	S
no vendor lock-in	workflow methodology should have broad industry support / multiple tool suppliers	S

tool/engine supports compliance measurements / task tracking / performance analysis	if a workflow methodology is used to model a protocol or SOP, then this a nice to have.	S
supports definition of time limits	method supports modeling time-based limits / deadlines / triggers of starting and completing tasks	M
can support monitoring processes as well as controlling, orchestrating them	different Philips products/solutions want to use workflow engines in different ways / at different levels - from monitoring, to advising, to controlling human tasks / activities	M
the graphical format can be used to show the progress of an individual case / execution of a flow	saves re-inventing / building another view on the progress of a case	S
supports separating responsibilities between solutions supplier and experts that define protocol/workflow	Philips most likely will not define workflows for clinical cases, but let others define SOPs etc. and model those.	M
supports roles / assignment of users to roles	Tasks can be assigned to roles and persons can have one or more roles. This must be covered by the methodology.	M
supports reporting of past/current case	for performance/adherence monitoring	S