Enhancing the HealthSuite Workflow Capability

EngD THESIS REPORT

Enhancing HealthSuite Workflow Capability: Optimizing Clinical Workflow Modelling and Integrating Virtual Patient

Tesfay Gebremeskel Chekole
October 2024
Department of Mathematics & Computer Science

EngD SOFTWARE TECHNOLOGY

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Enhancing the HealthSuite Workflow Capability: Optimizing Clinical Workflow Modelling and Integrating Virtual Patient

Tesfay Gebremeskel Chekole

October 2024

Eindhoven University of Technology
Stan Ackermans Institute – Software Technology

EngD Report: 2024/071

Confidentiality Status:
Public – open access

Partners

Philips HealthCare

Eindhoven University of Technology

Steering Group Ir. Bergevoet, Bas, PDEng
Ir. Patrick Bonne
Dr. Renata Medeiros de Carvalho, PhD
Dr. Yanja Dajsuren, PDEng

Date October 2024

Composition of the Thesis Evaluation Committee:

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

The design that is described in this report has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct.

Contact Address
Eindhoven University of Technology
Department of Mathematics and Computer Science
MF 5.080A, P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands
+31 402474334

Partnership
This project was supported by Eindhoven University of Technology and Philips.

Published by
Eindhoven University of Technology
Stan Ackermans Institute

EngD-report
2024/071

Preferred reference
Enhancing the HealthSuite Workflow Capability: Optimizing Clinical Workflow Modelling and Integrating Virtual Patient.
Eindhoven University of Technology, EngD Report 2024/071, October 2024

Abstract
This report aims to enhance the HealthSuite Workflow Capability (HSWC), a standardized software solution developed by Philips Healthcare, by optimizing clinical workflow modeling and integrating a Virtual Patient (Physiological Model Software) to create a more realistic testing environment. The research explores various workflow modeling approaches to identify the most effective methods for optimizing instrumented clinical workflows. Key goals include improving instrumented clinical workflow modeling to enable accurate and timely access to patient data, automatically closing User Tasks based on available data, and integrating the HSWC—particularly the FHIR store—with the Pulse Physiology Engine to simulate realistic patient conditions. The results demonstrate that enhanced workflow modeling and the integration of the HSWC with a Virtual Patient can significantly improve clinical workflow management, leading to better patient outcomes and operational efficiency.

Keywords
FHIR, Virtual Patient, Workflow modelling, Instrumented clinical workflow

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?

To address the challenges of clinical workflows, Philips has initiated an open-sourced HealthSuite Workflow Capability. This is a generic software component that applies the open industry standards BPMN and FHIR. The HealthSuite Workflow Capability has already existed for some time, however from experience we know that there are some desired extensions and improvements.

The work described in this thesis improves the HealthSuite Workflow Capability by leveraging the expression power of additional BPMN model constructs and by adding a means to exercise and demonstrate clinical workflows during development and testing of clinical applications. The improvements are validated and visualized by a demonstrator, enabling the open-source community to explore and improve further on these concepts.

We like to thank Tesfay for his contribution to the HealthSuite Workflow Capability, a next step in Philips's mission to improve people's health and well-being through meaningful innovation.

Bas Bergevoet / Patrick Bonné
September 2024

Preface

This report summarizes the project entitled "Enhancing HealthSuite Workflow Capability: Optimizing Clinical Workflow Modelling and Integrating Virtual Patient," which is part of the Engineering Doctorate (EngD) program in Software Technology at Eindhoven University of Technology.

Conducted in collaboration with Philips HPM, the project aimed to enhance the HealthSuite Workflow Capability (HSWC) by exploring various workflow modelling approaches and integrating a Virtual Patient to create a more realistic testing environment.

The report includes the following sections:

- Problem Analysis: This chapter outlines the challenges, objectives, research questions, and scope of the project.
- Requirements and Use Cases: Here, we detail the functional and non-functional requirements, along with relevant use cases.
- System Design and Architecture: This chapter describes key components, including the Physiological Model Software, Workflow Capability, and BPM Engine.
- Implementation: This chapter discusses the development process, enhancements made to the HSWC, and integration with the Virtual Patient or Physiological Model Software.
- Verification & Validation: This chapter covers the methods employed to ensure the system's reliability and effectiveness.
- Conclusions: This chapter presents the project results, lessons learned, and recommendations for future work.
- Stakeholder Analysis: This chapter addresses the roles and concerns of key stakeholders involved in the project.
- Project management: This chapter outlines the project management practices utilized throughout the project.

Recommended reading:

- Technical readers: System Design and Architecture (chapter 4) and Implementation(chapter 5)
- Healthcare professionals: Problem Analysis (chapter 2) and Requirements (chapter 3).
- Non-technical readers: Chapters 1, 2, 3, 8, and 9 for a comprehensive overview.

Tesfay Gebremeskel Chekole
October 2024

Acknowledgements

The work presented in this report would not have been possible without the invaluable support and guidance of many individuals around me. I would like to take this opportunity to express my sincere gratitude to all those who have contributed to my journey.

First and foremost, I want to thank my company supervisors, Bas Bergevoet and Patrick Bonné. Your supervision, guidance, and oversight of my progress have been essential. At the beginning of the project, I was unfamiliar with some of the concepts, including business process modelling and healthcare standardization. I truly appreciate the time and effort you invested from start to finish. I have gained so much from our discussions and the knowledge you shared. This work would not have been achievable without your contributions.

I would also like to extend my thanks to my TU/e supervisor, Renata Medeiros de Carvalho. Your assistance throughout the project has been invaluable. Your expertise in business process management and architecture has greatly aided my understanding and development.

I am grateful to Philips Healthcare, particularly the Software HPM team, and especially to Ralph Maesen, who guided me in selecting open-source tools like the Physiological Model Software. I appreciate the openness and approachability of the entire Philips HPM team; I was always welcomed for coffee and lunch with warm hospitality. Additionally, I want to thank the Philips intern students for fostering a friendly and safe working environment.

I would also like to acknowledge the Software Technology EngD management, coaches, and colleagues. In particular, I am thankful to Yanja Dajsuren and Karin Major for their support and guidance throughout the project and the program as a whole.

Lastly, I want to express my gratitude to my friends, both near and far. Your support and encouragement have been a tremendous help throughout the program, especially during challenging times when connecting with family was difficult for many Tigrayans around the world. Your advice, love, and encouragement have meant so much to me.

A special thank you goes to my family, who have always believed in me. Without your faith, support, and unwavering motivation, I would not have pushed my boundaries and achieved as much as I have.

Tesfay Gebremeskel Chekole
October 2024

Executive Summary

In today's healthcare landscape, accurate execution of clinical workflows is vital for optimal patient outcomes. Patients navigate multiple clinical pathways during their hospital stay, adhering to workflows established by large healthcare organizations, which are often locally adapted. To enhance this process, Philips Healthcare developed the HealthSuite Workflow Capability (HSWC), a standardized software solution that guides healthcare professionals in executing these workflows effectively.

This report presents the findings and outcomes of a project aimed at enhancing the HealthSuite Workflow Capability (HSWC) by optimizing clinical workflow modelling and integrating Virtual Patient (Physiological Model Software) to HSWC to simulate realistic patient conditions. The research involved exploring various workflow modelling approaches to identify the most effective methods for streamlining clinical pathways. Key goals included optimizing clinical workflow modelling approach while ensuring they remain functional and effective.

A significant aspect of the project was the integration of the Pulse Physiology Engine with the HSWC. This integration allows for the simulation of realistic patient conditions and responses to clinical interventions, providing a more accurate testing environment for evaluating clinical workflows. The results demonstrate that enhanced workflow modelling and the integration of HSWC, particularly FHIR store, with Physiological Model Software can significantly improve clinical workflow management, leading to better patient outcomes and operational efficiency.

The methodology employed in this project included systematic project management practices, stakeholder engagement, and iterative development processes. The findings highlight the importance of continuous feedback and communication among team members and stakeholders to ensure alignment with project goals.

In conclusion, the project not only provides valuable insights into optimizing clinical workflows but also offers recommendations for future research and development in healthcare IT systems. The findings underscore the capability of HSWC and Virtual Patient to transform healthcare practices, leading to improved patient care and operational effectiveness.

List of Abbreviations

API	Application Programming Interface
BPMN	Business Process Model and Notation
BPM	Business Process Management
DMN	Decision Model and Notation
E2E	End to End
EngD	Engineering Doctorate
FHIR	Fast Healthcare Interoperability Resources
HSWC	HealthSuite Workflow Capability
HAPI	Healthcare Application Programming Interface
HL7	Health Level 7
PDEng	Professional Doctorate in Engineering
REST	Representational State Transfer
OMG	Object Management Group
UML	Unified Modeling Language
TQ	Time Query
TU/e	Eindhoven University of Technology
V&V	Verification and Validation
WFC	Workflow Capability
XML	eXtensible Markup Language

Contents

List of Figures

List of Tables

1.Introduction

1.1 Project Context

Philips, a key player in the healthcare market, is strategically positioning itself to meet the contemporary challenges of the industry. The company's approach is shifting from being a provider of healthcare systems to becoming a holistic healthcare solutions provider. This means that the company is not just providing traditional patient monitoring tools but is now offering integrated solutions that manage all aspects of patient care throughout their stay in a healthcare facility. This approach reflects Philips' commitment to enhancing the overall efficiency and effectiveness of healthcare delivery. These solutions exemplify Philips' commitment to integrating its products with the actual workflows of hospital staff such as nurses, to provide actionable support throughout the various stages of patient treatment. This solution-oriented approach not only aims to enhance the efficiency of clinical workflows but also improves the quality of patient care by providing a seamless integration of data and functionality.

In its Hospital Patient Monitoring (HPM) division, Philips has initiated the development of a standard software solution known as HealthSuite Workflow Capability (HSWC) [1]. This open-source platform leverages best practices from within Philips, as well as the broader software community, including contributions from standardization bodies like Object Management Group (OMG)[1] and Health Level Seven International (HL7)[2]. The software is designed to run configurable Business Process Model and Notation (BPMN) models (workflows) and interacts with any clinical software system using a standardized HL7 FHIR interface.

Philips HPM is also focusing on integrating Virtual Patient (Physiological Model Software) with HSWC. The goal of integrating the Virtual Patient, specifically the Pulse Physiology Engine, is to simulate realistic patient conditions and their responses to clinical interventions that are guided by the workflow system. This capability enables healthcare professionals to assess how various treatments and actions may influence a patient's vital signs, such as heart rate and respiratory rate, prior to applying them in a real patient environment.

1.2 Outline

The report begins with an **Introduction** that provides the project context and outlines the document's structure.

The **Problem Analysis** chapter explores a few challenges associated with HSWC and defines the project's objectives, research questions, and scope. This sets the foundation for the work that follows.
In the **Requirements and Use Cases** chapter, functional and non-functional requirements are outlined, along with various use cases that the HSWC must address. This chapter is crucial for understanding the system's intended functions.

[1] The Object Management Group® Standards Development Organization (OMG® SDO) is an international (27 countries), membership-driven (230+ organizations) and not-for-profit consortium. [12]

[2] Health Level Seven International®, or HL7, is a member-driven nonprofit organization dedicated to creating and maintaining standards that bridge the gap in healthcare technology. Our goal is to improve the accessibility, speed, safety, security, quality, and cost of the electronic heath information exchange. [13]

The **System Design and System Architecture** chapter details the system's overall structure, including key components like the Physiological Model Software, Workflow Capability, and BPM Engine (Workflow Engine) providing a comprehensive view of the design.

The **Implementation** chapter covers the development process, including environment setup, integration with the Pulse Physiology Engine, WFC enhancement, and workflow model creation. It also discusses specific implementation details like Message Boundary Events and FHIR queries.

The **Verification & Validation** chapter explains the methods used to ensure system reliability and effectiveness, including stakeholder-driven code reviews, unit testing, inter-component testing, and scenario-based validation. The chapter concludes with a summary of the results, demonstrating the system's readiness for deployment.

The **Conclusions** chapter summarizes the project's outcomes, lessons learned, and future recommendations.

The **Stakeholder Analysis** chapter identifies the roles and concerns of key stakeholders, highlighting their contributions and expectations. The **Project Management** chapter outlines the methodology, communication strategies, deliverables, project timeline, and risk management, providing insight into how the project was managed.

Finaly, the report ends with a **Glossary**, **Bibliography**, and **Appendices** containing supplementary information.

It is important to note that this report is an extension of the PDEng report [1].

2.Problem Analysis

2.1 Context

As highlighted in the introduction, Philips is evolving from a traditional healthcare systems provider to a comprehensive healthcare solutions provider. The HSWC has been developed as part of this transition—a standardized software solution aimed at enhancing clinical workflows. HSWC seeks to standardize interfaces for applications by adopting the HL7 FHIR standard to facilitate electronic healthcare information exchange and improve interoperability between systems. By integrating a Business Process Model (BPM) engine, HSWC allows for executing configurable BPMN workflows. This enables healthcare personnel to follow structured processes aligned with clinical protocols, thereby improving clinical outcomes by streamlining workflows and minimizing errors. Ultimately, HSWC's overarching goal is to enhance patient care quality.

2.2 HealthSuite Workflow Capability (HSWC)

The HSWC was developed based on a high-level conceptual reference architecture, as illustrated in Figure 2.1. It will function within the HealthSuite digital platform, utilizing BPMN and Decision Model and Notation (DMN) models to guide clinical workflows while interfacing with clinical software systems through a standardized HL7 FHIR interface. The high-level conceptual reference architecture is thoroughly detailed in the Exploration Note on HealthSuite Workflow Capability [2]. As depicted in Figure 2.1, it comprises several key components: the HSWC, FHIR Store, Standardized Actions, Data and Triggers Interface, and Software Application. The report “Standardized Software Solution Designed to Guide Clinical Workflows” [1] describes more about the design and implementation of HSWC.

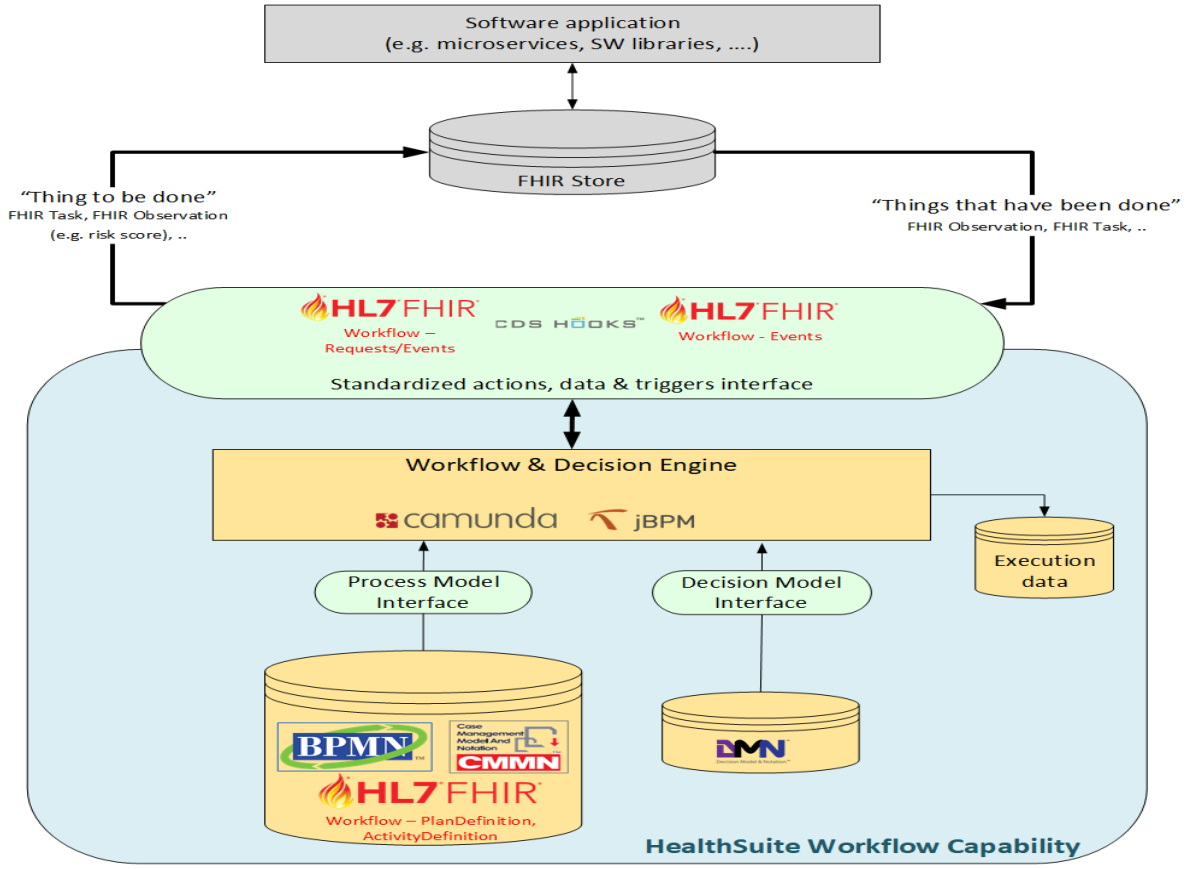

Figure 2.1: The HealthSuite Workflow Capability Reference Architecture [2]

In the HSWC, clinical workflows, as shown in Figure 2.2, require patient information stored in the local hospital's FHIR store to guide decision-making. The clinical workflows use a Data Object Reference[3] to show retrieval of data from the FHIR Store. A BPMN element Receive Task is employed to pause the workflow until the necessary data is available. Upon entering a Receive Task, a query is sent to the Workflow Capability, which communicates with the FHIR Store. Once the data is retrieved, a REST message is sent back to the Workflow Engine, where the data is stored and made available for use in the remainder of the workflow.

Despite its promising potential, the current system exhibits significant limitations that impede its effectiveness in real-world clinical applications.

One limitation of the HSWC is that that it does not offer a functionality to automatically close activities (User Tasks) in the BPM engine and update the task status to "completed" in the FHIR store when the required data becomes available. As a result, the activity (User Task) must be completed by a user from the application or the application itself and waits for a message to continue the workflow. In other words, as shown in Figure 2.2, the User Task “Measure sobriety” must be completed and the Receive Task “Get sobriety” comes after completion to get the data. That means there is no way to check if there is existing data that is valid for the workflow before the User Task is completed.

[3] In BPMN a **data object** represents information used or produced during a business process. It is an artifact that shows data flowing through a process, indicating what data is required or generated by an activity.

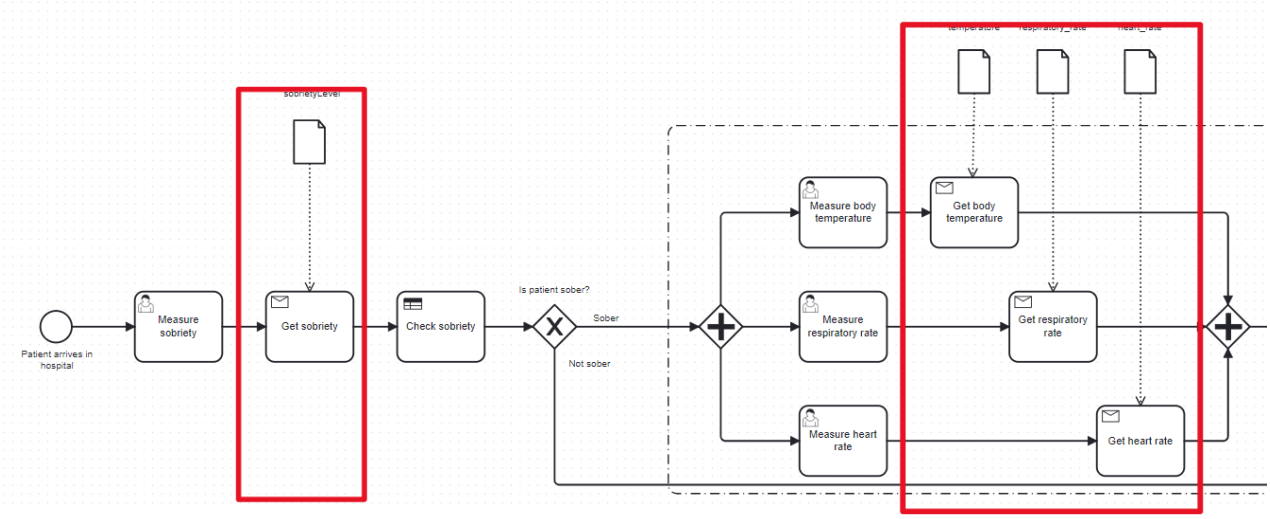

Figure 2.2: BPMN model of a simple sepsis protocol with Receive Task and Data Object Reference

Another limitation of the HSWC is that additional BPMN elements are needed to facilitate communication with external entities like the Workflow Capability. This can add unnecessary complexity and clutter to the diagram, making it verbose. In the sepsis protocol example shown Figure 2.2, the use of Receive Tasks and Data Object References clutters the visual representation of the clinical workflow, as highlighted in box. Exploring ways to abstract the model and replace these elements could simplify the workflow while maintaining its effectiveness.

Furthermore, as clearly described in the introduction, there is a pressing need to integrate a Virtual Patient to evaluate the clinical workflows in HSWC system with realistic patient conditions.

2.3 Project Goal and Objectives

2.3.1. Project Objectives

The goal of this project is to explore and identify an alternative solution for the existing workflow that reduces the number of elements in clinical workflow models, enables the automatic closure of activities based on available data, and integrating Virtual Patient to create a more realistic testing environment for the HSWC.

The specific objectives include:

- Exploring various approaches of modeling instrumented clinical workflows.
- Selecting the best modeling approach for creating an optimized instrumented clinical workflow
- Enhancing the HSWC to execute the optimized instrumented clinical workflows developed based on the selected approach.
- Integrating the Virtual Patient, the so-called Physiological Model Software, with HSWC

2.3.2. Research Questions

The project explored the following research questions to guide the development and implementation of the proposed enhancements:

- How can different instrumented clinical workflow modeling approach be explored and evaluated to identify the most effective alternative that reduces the number of elements while maintaining functionality?
- Can the Workflow Capability (WFC) automatically close activities in FHIR store when the required data becomes available, and what specific criteria must be established for this functionality to operate successfully?
- How can the physiological model software be integrated with the HealthSuite Workflow Capability (HSWC) to simulate realistic patient vital signs to test clinical workflows?

2.3.3. Scope

- Investigation and assessment of various clinical workflow modeling methodologies using the sepsis protocol as a case study (previously developed by Philips HPM)
- Enhancing HSWC that executes clinical workflow models developed based on the selected modeling approach.
- Developing an application that integrates Pulse Physiology Engine and FHIR store to simulate heart rate and respiratory rate.

3.Requirements and Use Cases

This chapter details the functional and non-functional requirements aimed at enhancing the HSWC for this project. These requirements focus on improvements to existing components, such as the Workflow Capability (WFC), Caregiver App, and BPM Engine, as well as the addition of new components, like the Virtual Patient. Additionally, relevant use cases and scenarios for these components are outlined to support the overall objectives of the project.

3.1 System Requirements

Through a review of stakeholder documents and regular meetings, the following functional and non-functional requirements have been identified.

3.1.1. Functional requirements

The functional requirements for the project are summarized in Table 3.1, organized by the MoSCoW **[3]** prioritization method, which categorizes each requirement as Must Have, Should Have, Could Have, or Won't Have.

Table 3.1 Functional requirements

Require-ment ID	Functional Requirement Description	MoSCoW
FR1	The HSWC shall support instrumented clinical workflows that are designed to either wait for data or proceed when data is not required.	M
FR2	The HSWC shall support automatic completion of User Tasks in FHIR store when data is available.	
FR3	The HSWC shall integrate Virtual Patient to simulate realistic vital sign of a patient.	M
FR4	The components in the HSWC shall communicate data to and from through the standardized HL7 FHIR interface.	M
FR5	The HSWC shall run previously developed instrumented clinical workflows .	M
FR6	The instrumented clinical workflow in HSWC shall support fetching data based on specific datetime from FHIR store.	M
FR7	The instrumented clinical workflows should support the combination of GET and SUBSCRIBE.	M

3.1.2. Non-Functional Requirements

The non-functional requirements for the project are summarized in Table 3.2, organized by the MoSCoW [3] prioritization method, which categorizes each requirement as Must Have, Should Have, Could Have, or Won't Have.

Table 3.2 Non-functional requirements

Requirement ID	Non-Functional Requirement Description	MoSCoW Priority
NFR1	The instrumented clinical workflow models shall be clear and easily understandable by users to facilitate quick adoption and minimal training.	**M**
NFR2	The HSWC shall be easily integrated with other systems to ensure it can easily connect with existing hospital and clinical information systems.	**M**
NFR3	The HSWC shall be capable of running in a Docker container to ensure consistency across different development and production environments.	M
NFR4	The developed instrumented clinical workflow models shall comply with BPMN standards to ensure best practices in workflow management are followed.	**M**
NFR5	The HSWC shall be engine or vendor independent to prevent vendor lock-in and allow flexibility in system implementation.	**M**
NFR6	The WFC in the HSWC shall only communicate with the FHIR store for clinical data communication.	**M**
NFR7	The enhanced HSWC shall contribute to a GitHub open-source repository.	M
NFR8	The instrumented clinical workflow models created in HSWC shall use fewer number of elements than the existing instrumented clinical workflows	M
NFR9	The HSWC shall allow parallel task execution	M

These non-functional requirements are designed to ensure that the HSWC is flexible, standards-compliant, and compatible with various clinical environments. This structure facilitates clear prioritization and focus during the development process.

3.2 Use Cases and Scenarios

To better understand and define the system requirements, we employ a series of use cases and scenarios.

3.2.1. Enhancing Workflow Capability

Use Case:
Optimize the existing instrumented clinical workflows by removing unnecessary Receive Task elements, creating a cleaner and more concise workflow diagram. Enhance the WFC to automatically check for available data in the FHIR store and complete task status.

Scenario:
In the HSWC, the FHIR store is updated with new laboratory results. The enhanced WFC

automatically detects the new data in the FHIR store and changes the task status to complete and facilitate automation for the User Tasks in the instrumented clinical workflow.

3.2.2. Accessing Data at Specific Time Points

Use Case:
Enable the WFC to access data at specific time points, providing flexibility when dealing with expired patient data.

Scenario:
An instrumented clinical workflow requires a patient's historical temperature readings for a specific date range. The WFC, with its ability to access data at specific time points, retrieves the relevant information from the FHIR store, even if some readings are expired or unavailable. This ensures the clinical workflow has the necessary data for informed decision-making in the process.

3.2.3. Parallel Task Execution

Use Case:
Develop a workflow that can execute parallel tasks, such as continuously measuring and sending heart rate values while performing other tasks.

Scenario:
During a stress test, a patient's heart rate needs to be continuously monitored while the healthcare provider records blood pressure and oxygen saturation. The parallel task execution feature of the instrumented clinical workflow allows for seamless monitoring of the patient's heart rate while simultaneously carrying out other necessary tasks, ensuring comprehensive patient care and efficient workflow process.

3.2.4. Integration of Physiological Model Software with Workflow Capability

Use Case:
Enable a clinical workflow to access realistic vital signs from a Virtual Patient during the execution of instrumented clinical workflow models.

Scenario:
During a treatment protocol, the Virtual patient provides patient data (e.g., heart rate, respiratory rate). The Workflow Capability uses this data for treatment protocols, ensuring that patient care decisions are based on realistic and up-to-date information.

Use Case:
Enable continuous monitoring and interaction between a Virtual Patient (via Physiological Model Software) and the FHIR store within the HSWC.

Scenario:
During a clinical pathway, the Virtual Patient (using Physiological Model Software) continuously provides vital signs such as heart rate, and respiratory rate to the FHIR store within the HSWC. The healthcare provider can monitor these real-time data points during the workflow. If the data indicates a critical change in the patient's condition, the system shows the nurse or doctor, allowing for immediate adjustments to the treatment plan based on the updated information, ensuring patient safety.

4. System Design and Architecture

This section introduces the design decisions, architectures and necessary diagrams that describe the overall system and the areas where the HSWC has been improved and new components are integrated into the HSWC.

4.1 Workflow model

The workflow model is a visual representation of a process that helps organizations simplify their operations, enhance efficiency, and achieve better outcomes. It provides a clear understanding of how tasks are executed, their flow, dependencies, and overall structure, facilitating process analysis, improvement, and standardization. By modeling a workflow, industries including healthcare can identify areas for improvement, automate tasks, and integrate with technology systems, ultimately leading to increased productivity and reduced manual effort [4].

Figure 4.1: Workflow model representation of two tasks using BPMN

As shown, Figure 4.1 represents a simple workflow model with two sequential tasks, Task A followed by Task B, starting from an initial point and ending at a final point.

4.1.1. Clinical workflow

Clinical workflow refers to the sequence of tasks, processes, and activities performed by healthcare professionals to deliver patient care. Clinical workflow management is essential for improving efficiency and productivity in healthcare settings. Workflow engines can help automate clinical processes by orchestrating tasks, managing data flow, and coordinating activities across different systems and personnel.

BPMN is a widely used standard for modeling clinical workflows, providing a visual representation of processes that can be easily understood by both technical and non-technical stakeholders [5]. Meanwhile, HL7 FHIR (Fast Healthcare Interoperability Resources) complements workflow management[4] by enabling standardized data exchange between different healthcare systems. FHIR's PlanDefinition resource can be used to define clinical workflows, while its Task resource supports the execution of workflow steps [1]. By combining workflow engines, BPMN modeling, and FHIR-based data exchange, healthcare organizations can create more efficient, interoperable clinical processes that improve patient care and outcomes [6] [7].

4.1.2. Instrumented Clinical Workflow models

Instrumented clinical workflow models are enhanced workflows that incorporate specific elements and instructions. These are executed by a workflow engine to ensure seamless clinical processes. In HSWC,

[4] Clinical workflow management is the process of designing, executing, monitoring, and optimizing the sequence of tasks and activities required to deliver high-quality patient care and achieve specific healthcare outcomes.

these models are designed to interact with the WFC, allowing access to clinical data from the FHIR store. For example, they can retrieve vital signs, which are crucial for informed decision-making within the clinical workflow. These models play a critical role in HSWC by enabling more efficient, data-driven processes, ultimately improving patient care outcomes.

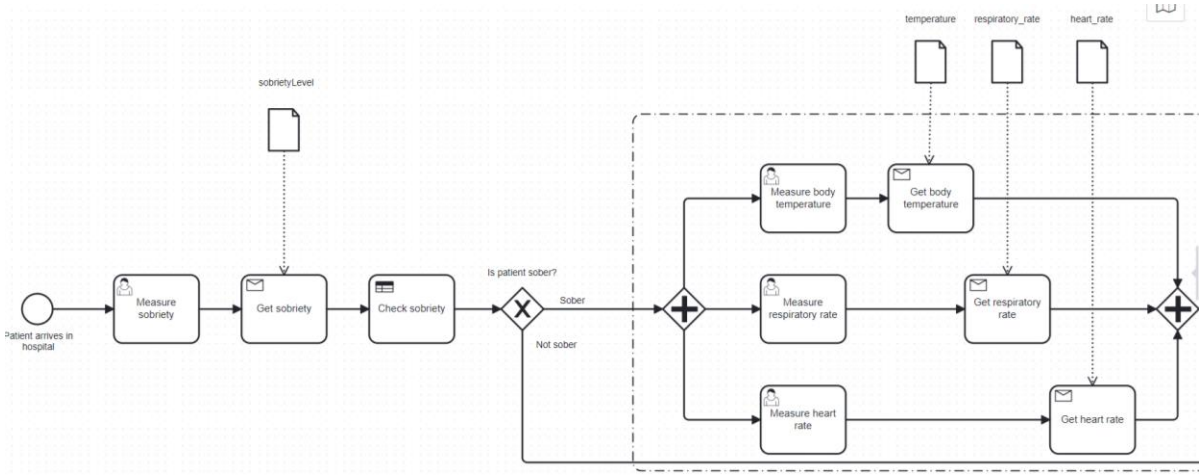

Figure 4.2 Instrumented clinical workflow sample for sepsis protocol

In Figure 4.2, the "Get sobriety" Receive Task and the "sobrietyLevel" Data Object Reference were added to facilitate communication with the Workflow capability component in HSWC. The instruction to communicate with the WFC in HSWC is set in the Data Object Reference and Receive Task [1].

4.1.3. Workflow Model Exploration and Alternative Modeling Selection

Section 4.1.2. discussed adding elements to communicate with WFC component in HSWC to obtain important data from FHIR store for decision-making. However, as shown in Figure 4.2, adding a Receive Task attached to a Data Object reference can create verbosity and clutters the message it delivers in the workflow. To address this and other functional and non-functional requirements of the system, several alternative instrumented clinical workflow models were evaluated to identify the most suitable approach. The alternatives considered include the Receive Task with Data Object Reference as a baseline, User Task with Data Object Reference, Message Boundary Events on User Task, and Message Intermediate Catch Events and are explored in this report. For this report, we used Sepsis protocol to instrument and experiment our work. So, sample design clinical workflow (Sepsis protocol as use case) for each alternative is presented in **Appendix A**. To systematically compare these alternatives, a Pugh matrix is employed based on several criteria, derived from both functional (FR) and non-functional (NFR) requirements, outlined in the requirements section. The criteria derived from the requirements define how we evaluate if the system meets the requirements, ensuring that the chosen instrumented clinical workflow modeling approach effectively addresses the project's objectives. These criteria include clarity of workflow, optimized workflow model creation , data-independent flow continuation, data-driven flow continuation, alternative flow availability, and parallel execution.

To make the comparison clear and easy to understand, we linked each evaluation standard (or "criterion") to the specific requirement it comes from.

- Clarity of Workflow (derived from NFR1: The workflow models should be clear and easily understandable)
- Optimized Workflow Model Creation (derived from NFR8: The instrumented clinical workflow models should use fewer elements than existing instrumented clinical workflows),

- Data-Independent Flow Continuation (derived from FR1: The HSWC shall support instrumented clinical workflows that are designed to proceed when data is not mandatory)
- Data-Driven Flow Continuation (derived from FR1: The HSWC shall support instrumented clinical workflows that are designed to wait for data)
- Alternative Flow Availability (derived from FR1: The HSWC shall support instrumented clinical workflows that are designed to either wait for data or proceed when data is not required)
- Parallel Task Execution (derived from NFR9: The HSWC should support parallel task execution).

The Pugh matrix is a decision-making tool that compares multiple alternatives against a baseline option. Its defining feature is the use of relative scoring, not absolute values. This approach allows for a clear comparison of strengths and weaknesses across different options. In the context of evaluating workflow model alternatives:

- "+" (Plus): Indicates a relative advantage or better performance compared to the baseline in a specific criterion.
- "0" (Zero): Signifies performance equal to the baseline, with no significant difference.
- "-" (Minus): Denotes a relative disadvantage or poorer performance compared to the baseline in that criterion.

It is important to understand that these symbols represent relative comparisons, not absolute scores. Each alternative was evaluated in direct relation to the baseline for each criterion. This relative scoring is the key characteristic of a Pugh matrix, allowing for a nuanced comparison that highlights differences between options. The results are summarized and presented in Table 4.1, which provides a Pugh matrix comparison with the instrumented workflow.

Table 4.1 Pugh matrix comparison with instrumented workflow

Criteria	Receive Task with Data Object Reference (base) (A)	User Task with Data Object Reference (B)	Message Boundary Events on User Task (C)	Message Intermediate Catch Events (D)
Clarity of Workflow	0	-	0	0
Optimized Workflow Model Creation	0	+	+	+
Data-Independent Flow Continuation	0	+	+	0
Data-Driven Flow Continuation	0	-	0	0
Alternative Flow Availability	0	0	+	0
Parallel execution	0	0	+	0
Net score	0	0	4	1

Criteria:

1. **Clarity of workflow:** This criterion evaluates if the workflows are easy to understand. Approach C and Approach D use a similar way of conveying information comparing to the Base approach (A), with the help of message symbol which helps to clearly understand that message is coming. The User Task with Data Object Reference (B) approach in Table 4.1 lacks the message symbol, and so it scores a "-" for its lower clarity comparing to the base approach.
2. **Optimized Workflow Model Creation:** We aimed to determine if the models could be simplified to use fewer elements than the Base approach. The core assumption of this criterion is that fewer elements are used to construct the instrumented clinical workflow compared to the Base approach. However, it's crucial to carefully consider which elements can be removed without compromising important information. This is not considered in this criterion. Thus, all approaches use fewer elements than the base approach.
3. **Data-Independent Flow Continuation:** In this criterion, we wanted to check if the workflows allow for the process to proceed even if there is no data in the FHIR store. In User Task with Data Object Reference approach (B), there is no way to force to wait for data from FHIR store, which means that a manual completion of the task allows it to continue without data available. The Message Boundary Events on User Task approach (C) allows for continued flow through the design of alternative paths. However, Option D is like the base approach which needs to continue only once data is available.
4. **Data-Driven Flow Continuation:** We wanted to make sure the workflows can be forced to only move forward when there was data available. The Message Boundary Events on User Task approach (C) can wait until the required data is available. The Message Intermediate Catch Event approach (D) is designed to wait for a message, which necessitates remaining in the workflow until the appropriate data arrives, like the Base Approach (A). However, in the approach using Data Object Reference with User Task (B), it is not possible to force waiting for data, as it is possible for the task to be completed manually.
5. **Alternative Flow Availability:** This criterion assesses the presence of alternative pathways or branches within the workflow to accommodate various scenarios or exceptions. The Message Boundary Events on User Task approach (C) is unique in its ability to support two flows from a single activity (one for when data is available and another for when data is not available).
6. **Parallel Task Execution:** We wanted to know if the system could allow the process to continue when data is available, while still waiting for new data. For example, continuously receiving patient heart rate value while measuring temperature at the same time. Only Message Boundary Event on User Task Approach (C) supports this feature because the Message Boundary Events can be configured as non-interrupting, providing flexibility in handling different situations [8] [9]. In section 4.1.4. detailed information about non-interrupting Message Boundary Event is presented.

Each alternative was evaluated against these criteria, providing a thorough analysis of their strengths and weaknesses relative to the current way of constructing instrumented clinical workflow models using Receive Task and Data Object Reference.

Design decision: *Following the analyses from the Pugh matrix (as shown in Table 4.1), and discussions with stakeholders the "Message Boundary Events on User Task" approach has been selected as the best alternative approach.*

4.1.4. Workflow Modeling with Boundary Message Event

In BPMN, a Message Boundary Event is an intermediate event that is attached to the boundary of an activity. It monitors specific messages during the execution of that activity.

Interruptive Message Boundary Event: as shown in Figure 4.3, the workflow pauses at the User Task A until the required data is available, after which the token proceeds to the next activity which is User Task B. In this case, if there is no message coming, the workflow for the given patient is stuck. In the case of User Task B, if there is no message coming, there is an option to close the activity and proceed to the alternative path that is connected directly to the User Task.

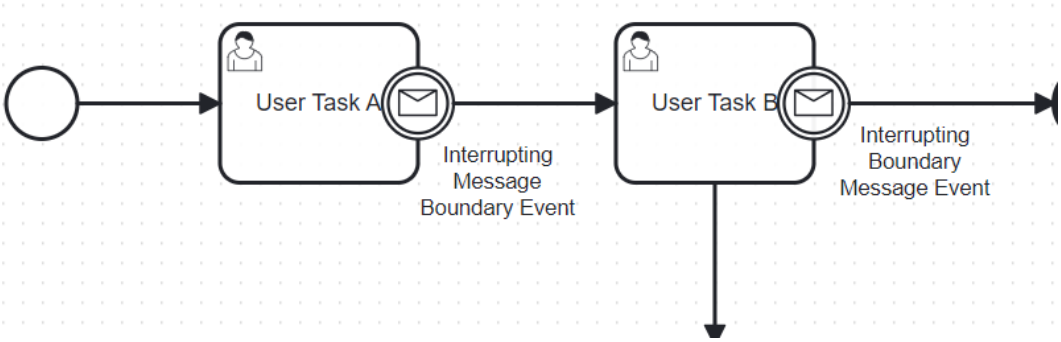

Figure 4.3: Message Boundary Event(interrupting) on User Task

Non-Interruptive Message Boundary Event: in the configuration as shown in Figure 4.4, the workflow continues along its designated path while waiting for the required data, thus facilitating parallel processing.

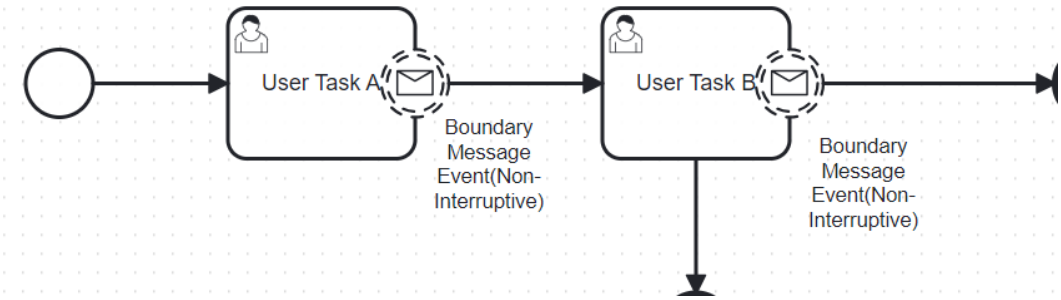

Figure 4.4 Message Boundary Event (non-interrupting)

In Figure 4.4, when the token enters User Task A, it waits for the corresponding message to arrive. When a message arrives the flow continues to User Task B without interrupting the activity User Task A, that means, the Message Boundary Event in User Task A can still accept messages and the activity itself is still active that allows parallel processing of activity. In the case there is no message in User Task A for the first time, then the workflow is stuck. User Task B has an alternative path in case there is no message arrive or end of activity. When stuck occurs due to the configuration or the design of the instrumented clinical workflow model in the system, it only affects that particular patient's care plan. This does not affect the entire system.

Interruptive and Non-Interruptive Message Boundary Events: as shown in Figure 4.5**,** the workflow utilizes both interruptive and non-interruptive Message boundary events. This kind of design can be used according to the behavior of the clinical workflow. For example, to measure temperature we can use one time activity in each workflow, that means once we measure the temperature next activity is followed (using the interruptive message boundary event). But for example, when used for heart rate, which might be important to continuously need the data while doing other activities, the non-interruptive message boundary event might be used.

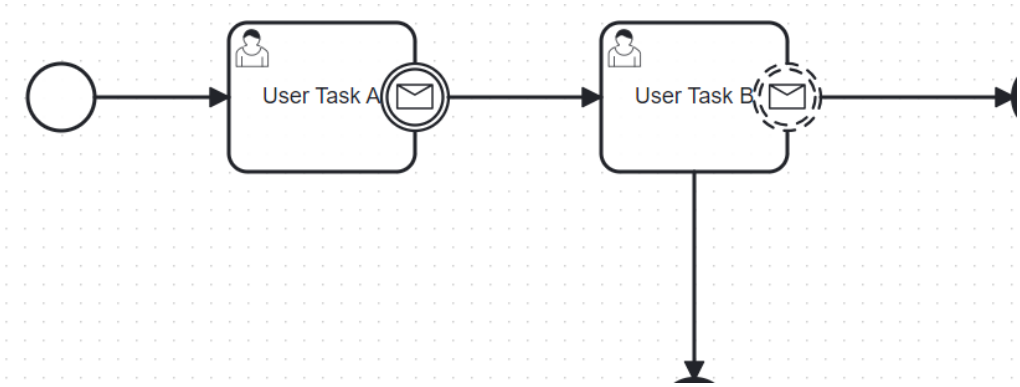

Figure 4.5: Interruptive and Non-Interruptive Message Boundary Events

Because different hospitals can use different clinical workflow processes, having different design options can be helpful and the system becomes robust and flexible.

Instrumented Clinical Workflow Model with Message Boundary Event, Sepsis use case

As presented in Figure 4.2, instrumented clinical workflow model sample for sepsis, Receive Task and Data Object Reference was utilized. In this section we presented the output of the same protocol using User Task and Message Boundary Event (interrupting) as shown in Figure 4.6.

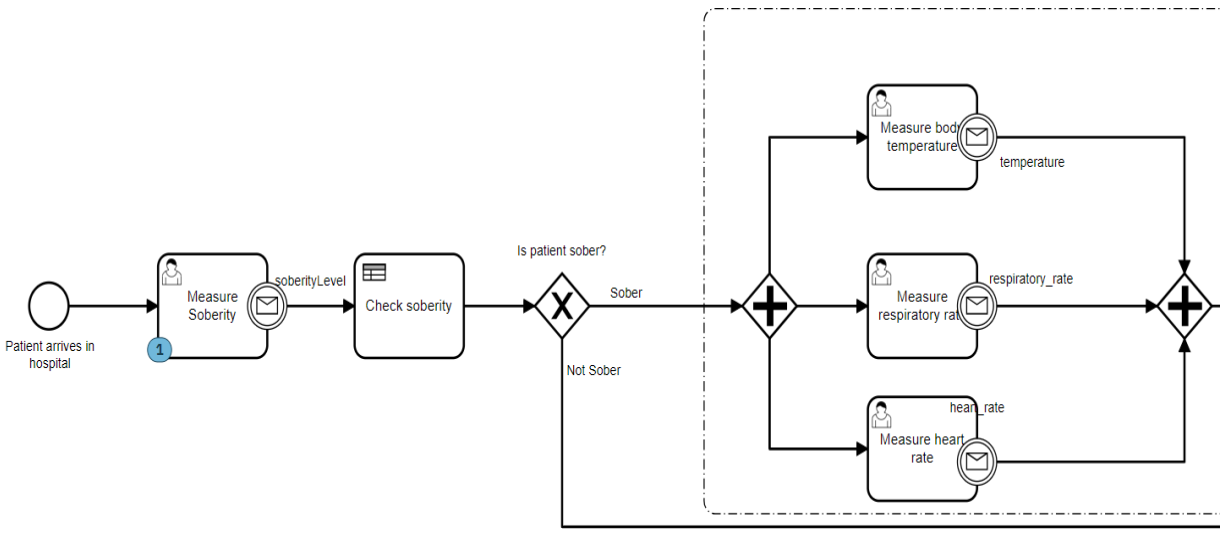

Figure 4.6 Instrumented Clinical workflow model developed based on Message Boundary Event

The newly selected clinical workflow modeling approach, utilizing Message Boundary Events, significantly optimizes the workflow compared to the existing model with Receive Tasks. By eliminating unnecessary elements and reducing visual clutter, the workflow becomes easier to understand. This

approach also provides greater flexibility in designing instrumented clinical workflow models, allowing for both interrupting and non-interrupting behavior as well as data-driven and data-independent features.

Design decision: *For the instrumented clinical workflow models, we introduce FETCH keyword in addition to the existing keywords GET and SUBSCRIBE in the query for communication with WFC. Then the WFC understands the keywords as follows:*

- *GET: checking data in the FHIR store if available and get it.*
- *SUBSCRIBE: get data when entered the database, this doesn't apply for the existing data in the FHIR store*
- *FETCH: checking data in the FHIR store if available get it if not SUBSCRIBE.*

In conclusion, the optimized workflow model using Message Boundary Events enhances clarity, flexibility, and efficiency (automatically data fetching feature) in clinical workflows. It's recommended to transition to this new approach for improved workflow management within the HSWC.

4.2 System Architecture

The general system architecture of the HSWC is designed to integrate various components that work together to enhance clinical workflows and patient care.

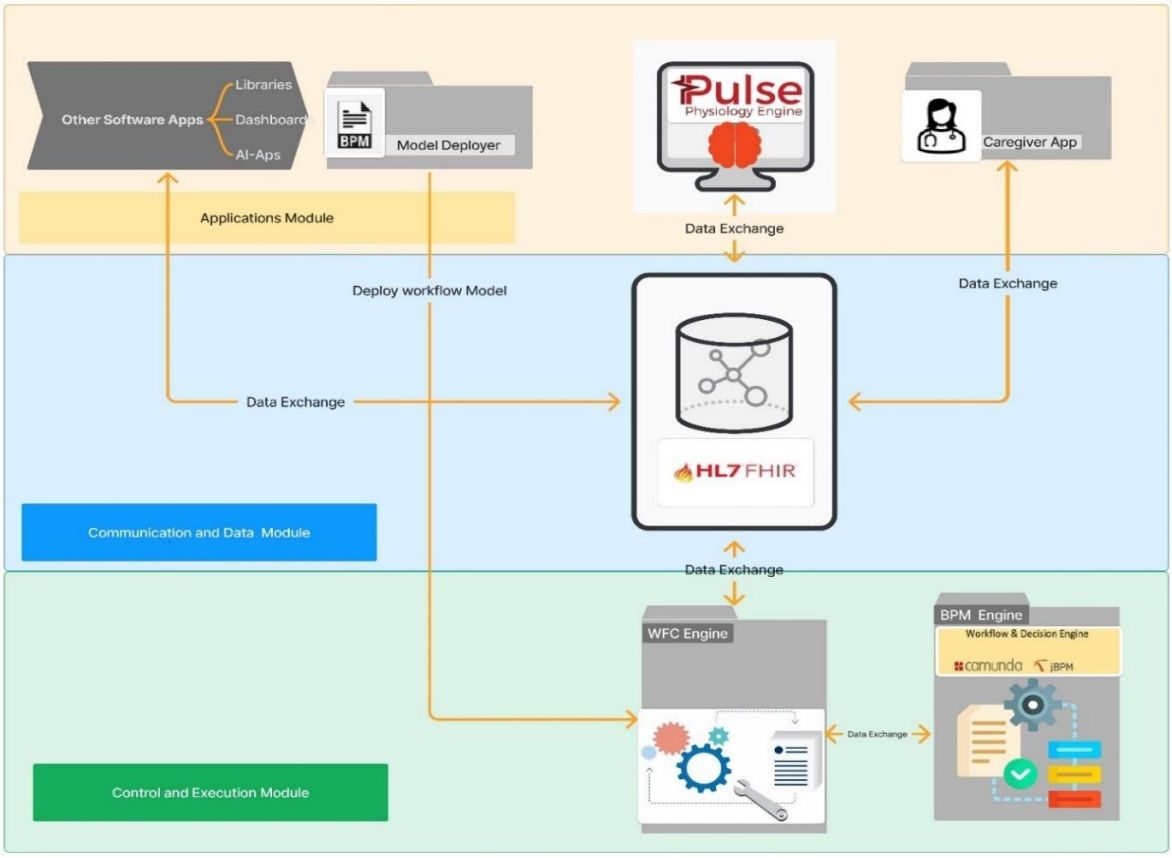

Figure 4.7 High-level System Architecture of HSWC

The high-level system architecture includes the applications module, communication and data module, and control and execution module. Those modules are described as follows.

4.2.1. Applications Module

This module includes various applications that healthcare providers use to interact with the HSWC, such as the Caregiver App and Physiological Model Software, and Other Software Applications.

4.2.2. Communication and Data Module

This module is responsible for facilitating communication between different components of the system.
HL7-FHIR: The HL7 FHIR standard facilitates the electronic exchange of healthcare information. It is widely adopted for its flexibility and interoperability, allowing different systems to communicate effectively.

4.2.3. Control and Execution Module

This module manages and controls the execution of clinical workflows. It includes WFC and BPM engine.
Workflow Capability: The WFC controls and manages workflow activities between the FHIR Store and BPM Engine.
BPM Engines: BPM engines or workflow engines are software tools designed to model, execute, and monitor business processes throughout their lifecycle. These engines help organizations automate and optimize workflows, enhancing efficiency and productivity.

Overall, the high-level system architecture illustrates how the components of HSWC interact to execute clinical workflows for specific protocols in hospitals. This system supports healthcare providers, such as nurses and doctors, to follow workflows and manage care plans.

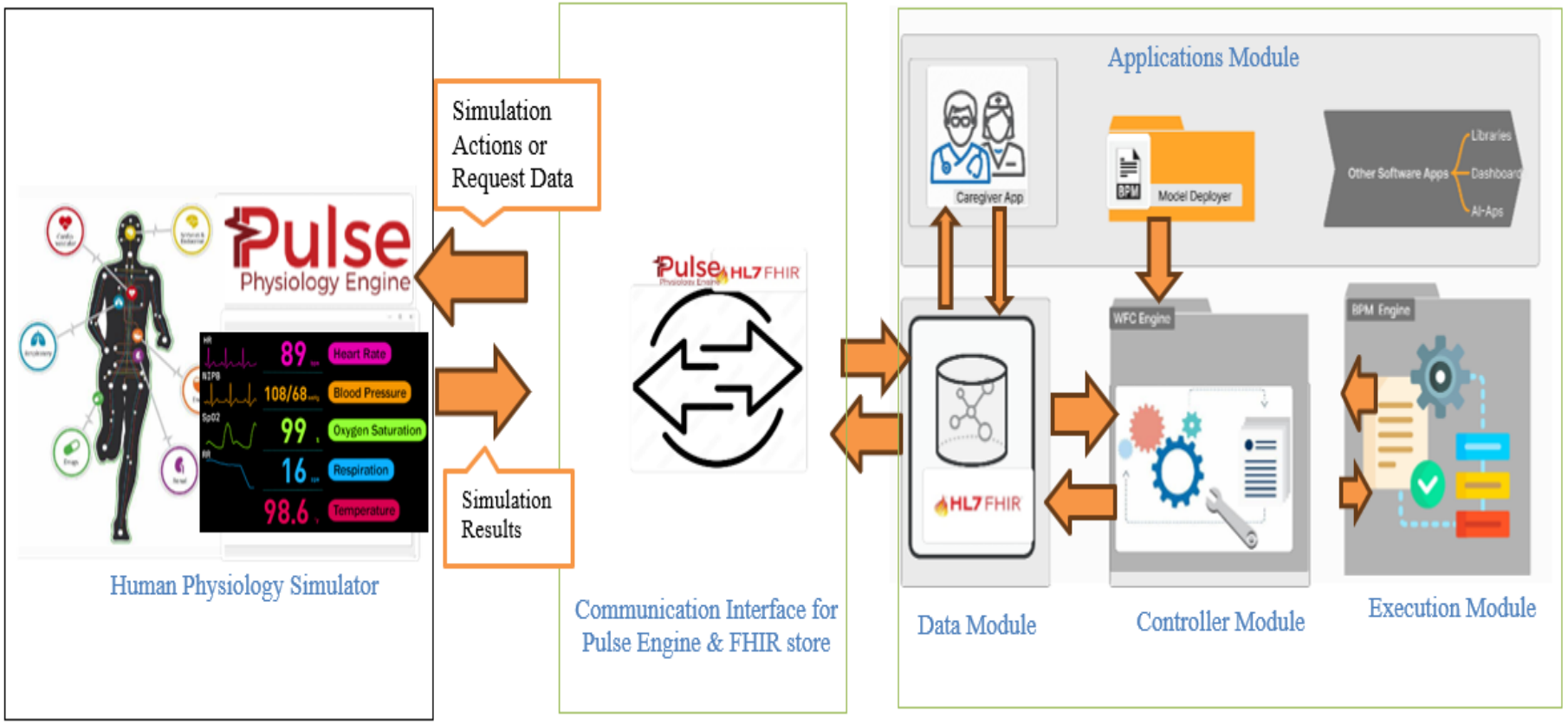

Figure 4.8 Simplified System Architecture for HSWC

Figure 4.8 expands the high-level architecture by focusing on the integration of the Virtual Patient (Pulse Physiology Engine) with the FHIR store. This integration allows healthcare professionals to observe how different treatments and actions might affect a patient's vital signs, such as heart rate and respiratory. By combining the HSWC with Virtual Patient, it helps to effectively predict patient outcomes, manage care plans, and improve decision-making. This capability is particularly useful for training purposes, testing treatment protocols, and improving the overall quality of healthcare delivery without relying on real patient scenarios.

4.2.4. Pulse Physiology Engine

The Pulse Physiology Engine is an open-source, multi-platform human physiology simulator developed in C++. It is compatible with Windows, Mac, and Linux, and serves as a comprehensive tool for medical education, research, and training technologies. This engine facilitates accurate and consistent simulations of human physiology within the medical community [10].

Design decision: *Although the stakeholders recommended integrating the Pulse Physiology Engine with the HSWC, we also investigated its compatibility with different programming languages, including Python, Java, C, and C++. We ultimately chose to integrate our system with Python through Jupyter notebooks, as it provides an API and clear documentation that facilitate integration with other systems, making it easier to use for both research and educational purposes.*

Pulse Engine to FHIR Interface App

Pulse Engine FHIR Interface App was introduced and designed to connect the Pulse Physiology Engine to FHIR Store to allow simulation actions and data requests from the Physiological Model Software (e.g., heart rate, respiratory rate). The results are returned to the Pulse Engine FHIR Interface App, enabling the interaction between the Virtual Patient and other HSWC components.

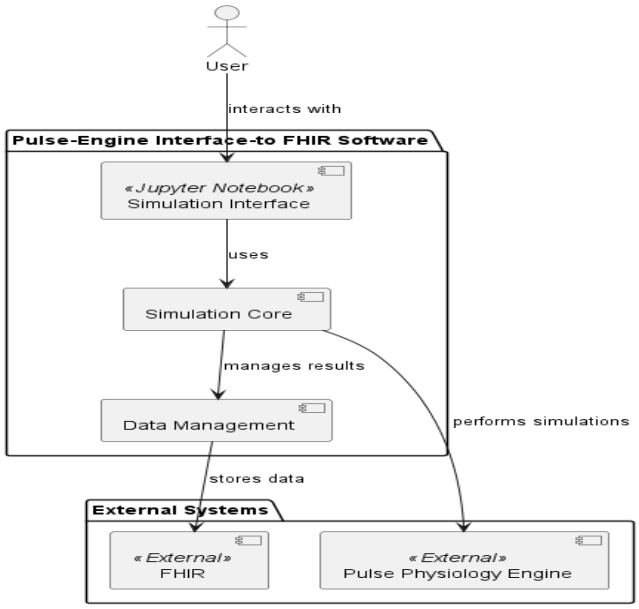

Figure 4.9 Pulse Engine to FHIR Interface App component diagram

The UML component diagram, as shown in Figure 4.9, shows the main components of the Pulse Engine FHIR Interface App by highlighting main components, such as the Simulation Interface, Simulation Core, and Data Management and their interaction with other external components like FHIR store and the Pulse Physiology Engine.

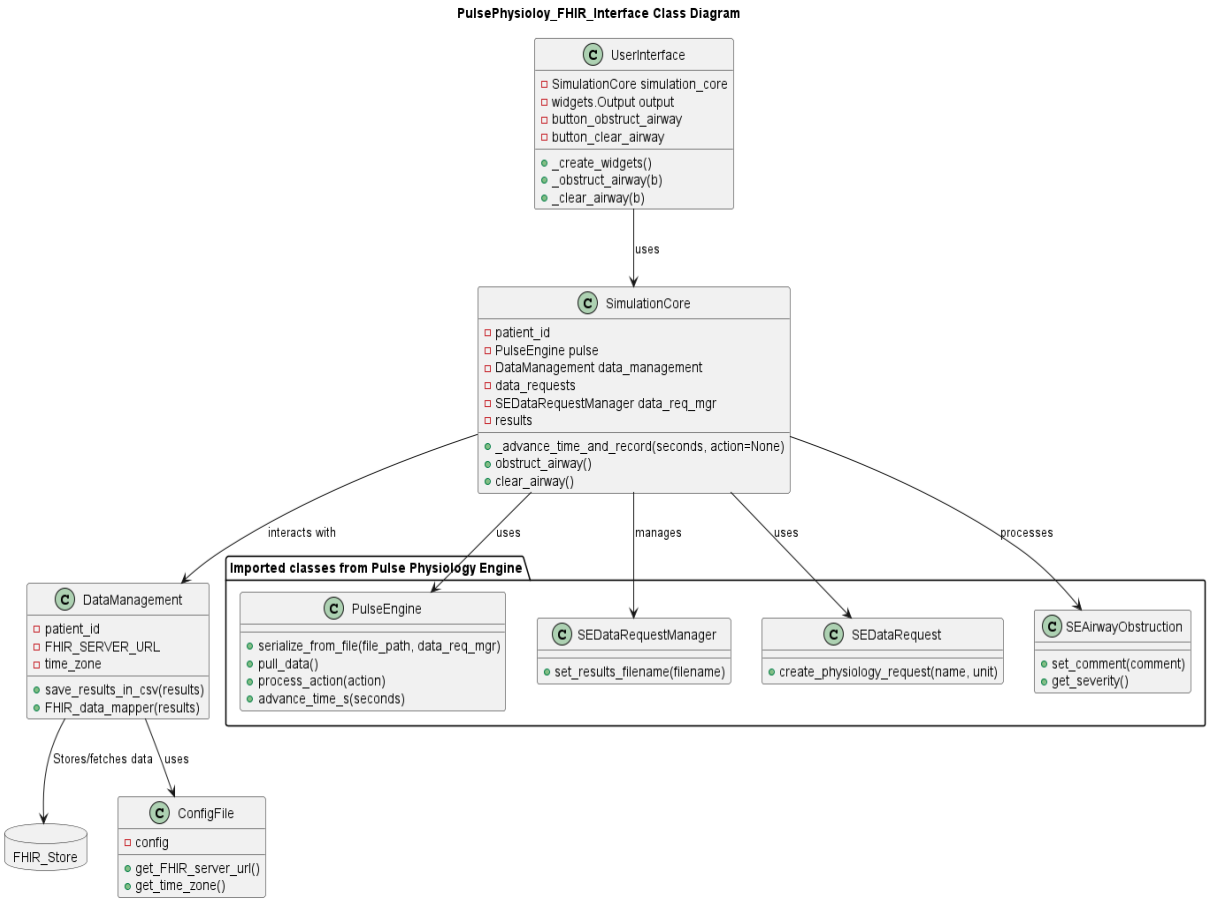

Figure 4.10: Pulse Engine-FHIR Interface App class diagram

Figure 4.10 illustrates the structural design of the Pulse Physiology Engine-FHIR Interface App that integrates Pulse Physiology Engine with HSWC particularly to the FHIR standard. The class diagram provides a detailed view of the various classes involved in the system and highlights how they interact to facilitate the simulation and management of patient physiological data.

Key Aspects of the Class Diagram:

1. **Classes**: The diagram identifies the key classes that make up the Pulse Physiology Engine and its integration with FHIR. Each class represents a specific component of the system, encapsulating its attributes and behaviors.
2. **Core Components**:
    - **User Interface Class**: This class provides a graphical interface through which users interact with the simulation. It includes functionalities for initiating simulations and displaying results.
    - **Simulation Core Class**: This is the central component of the simulation engine, responsible for managing the simulation timeline, generating physiological data, and coordinating interactions with other classes.
    - **Data Management Class**: This class handles the storage and retrieval of simulation data. It is responsible for mapping simulation results to FHIR resources, ensuring that data is formatted correctly for integration with FHIR servers.
3. **Integration with FHIR**: The class diagram emphasizes how the Pulse Physiology Engine interacts with the FHIR standard. It shows how patient data generated by the simulation is structured and stored in a way that complies with FHIR specifications, facilitating interoperability with other healthcare systems.

Overall, the class diagram serves as a blueprint for understanding the internal structure of the Pulse Physiology Engine and its integration with FHIR. It highlights the key components and their interactions, providing insights into how the system processes patient data and supports clinical workflow.
The code for the classes can be found in the repository Philips-labs/workflow-capability[5].

4.3 Workflow engine

As discussed in 4.2.2. as part of control and execution module, workflow engine (BPM engine) is a software/application that executes the clinical workflow model.
The main activities in Figure 4.11 are:

- **Send User Task to WFC:** this activity is recording the User Task into the FHIR store with status "ready" and well described in the sequence diagram Figure 4.12.
- **Process WFC Query**: in this activity, all process variables and time expressions were processed to the WFC accepted format. The format is represented in Figure 5.2.
- **Send Query to WFC**: after processing the query and formatted in WFC standard, the BPM engine sends the data to WFC via REST API. This activity is well presented in sequence diagram Figure 4.13.
- **Get data from WFC**: after sending the data, the BPM Engine (workflow Engine) waits for data from FHIR store, then the WFC process query and sends to FHIR store either to get data or to subscribe until data becomes available depending on the configuration of the query. This activity is also included in the sequence diagram Figure 4.13.

[5] workflow-capability/pulse-physiology_FHIR/ppe_fhir.ipynb at main · philips-labs/workflow-capability (github.com)

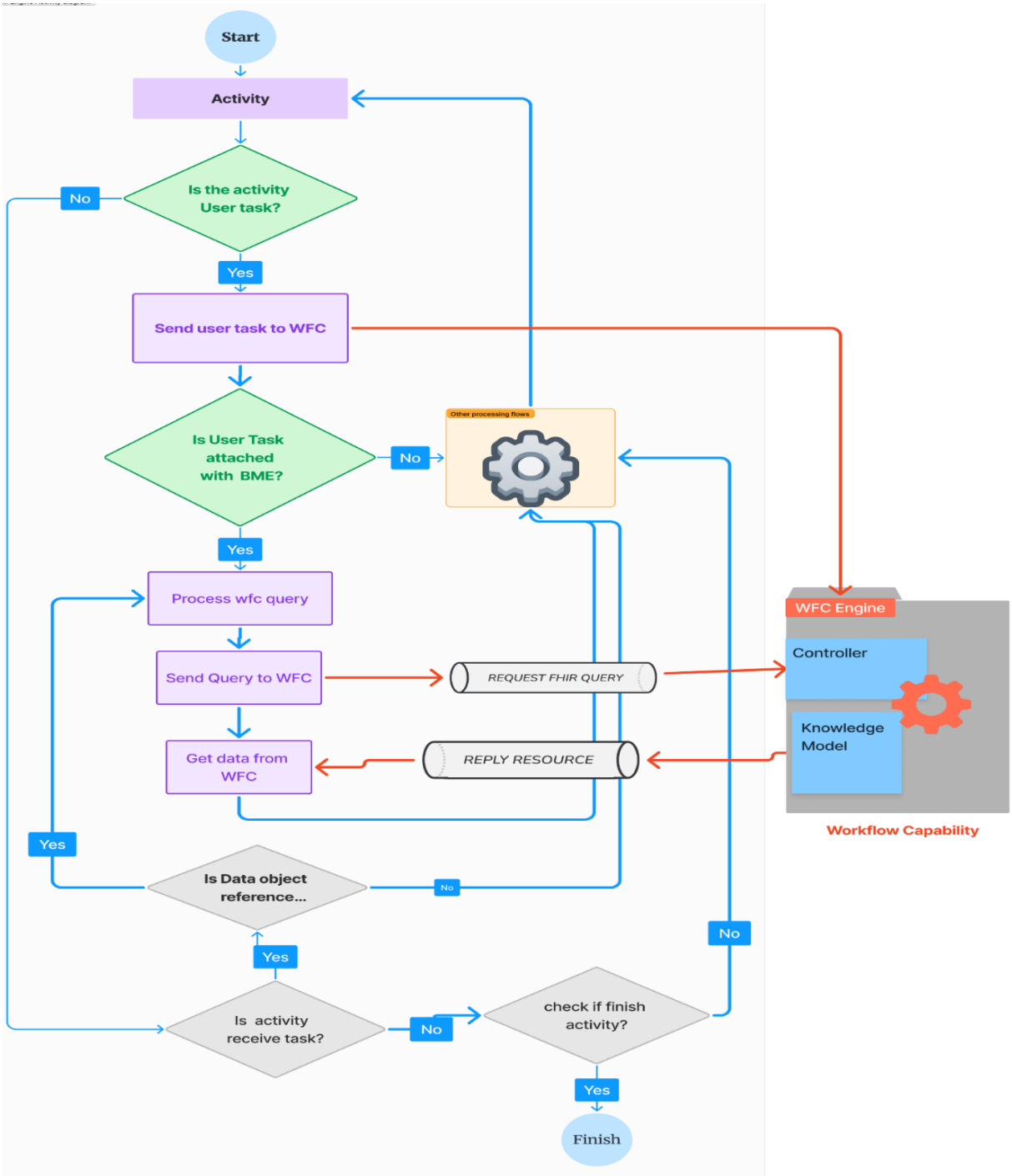

Figure 4.11 Activity diagram of BPM engine interaction with WFC

Figure 4.11 illustrates the activity diagram for BPM Engine or Workflow engine interaction during the execution of a clinical workflow models. It also shows the interaction between the BPM engine component and the workflow capability component (WFC Engine).

The diagram in Figure 4.11emphasizes the HSWC is supporting workflow models configured and instrumented with Message Boundary Event attached or with User Task and Receive Task with Data Object Reference to communicate with other components of HSWC via WFC, as shown in Figure 4.11.

Design decision: *The newly selected clinical workflow modeling approach or methodology, utilizing Message Boundary Events is alternative to the existing clinical workflow modeling approach which utilizes Receive Task with Data Object Reference attached to the Receive Task. That means that all the previously developed models using Receive Task are still valid and can run on the system. And in fact, the new WFC has the ability to run hybrid instrumented clinical workflow models (a model that constructs using Message Boundary Event and Receive Task with Data Object reference) for modeling flexibility.*

4.4 Workflow capability

The Workflow Capability includes control module that manages Knowledge Models and controls access to the Workflow Engine and FHIR Store, consisting of two main components:

1. **Knowledge Model Manager:** This component processes incoming Knowledge Models by converting XML definitions of BPMN and DMN into formats compatible with the FHIR Store and the specific Workflow Engine. In the report [1], section 5.4.3, Knowledge Model Manager is clearly explained within the workflow capability.
2. **Workflow Capability Core:** This part synchronizes the Execution Module and Data Module with the current workflow state and supports the Workflow Engine. It features a Subscription Controller that connects the FHIR Store and Workflow Engine, handling requests like updating workflow statuses.

To understand the overall design of WFC, refer to the report [1], section 5.4.
Once workflow is initiated and flows to User Task activity, the Workflow Engine interacts with WFC component and WFC interacts with FHIR Store to store the User Task activity in the database. Figure 4.12 shows the sequence diagram detailing how the User Task is stored in the FHIR Store.

Figure 4.12 Sequence diagram how User Task is stored in FHIR

The sequence diagram as shown in Figure 4.12 illustrates how User Task activity is created and helps to clarify how the User Task is executed, what information is required, and how components of the HSWC communicate with each other such as the BPM engine(Camunda[6]), WFC and the FHIR server. There is similar figure in the report [1], Figure 5.7, with name "Workflow Task Signaling - Sequence Diagram", however in this report, there is an additional component added: InterfaceWFCHandler and

[6] **Camunda**: An open-source platform that offers both community and enterprise editions. It's known for its powerful BPMN modeling capabilities and scalability [17].

improvement of the function in the end point method **PostRequestUserTask(instance, taskDefinitionKey, taskId).**

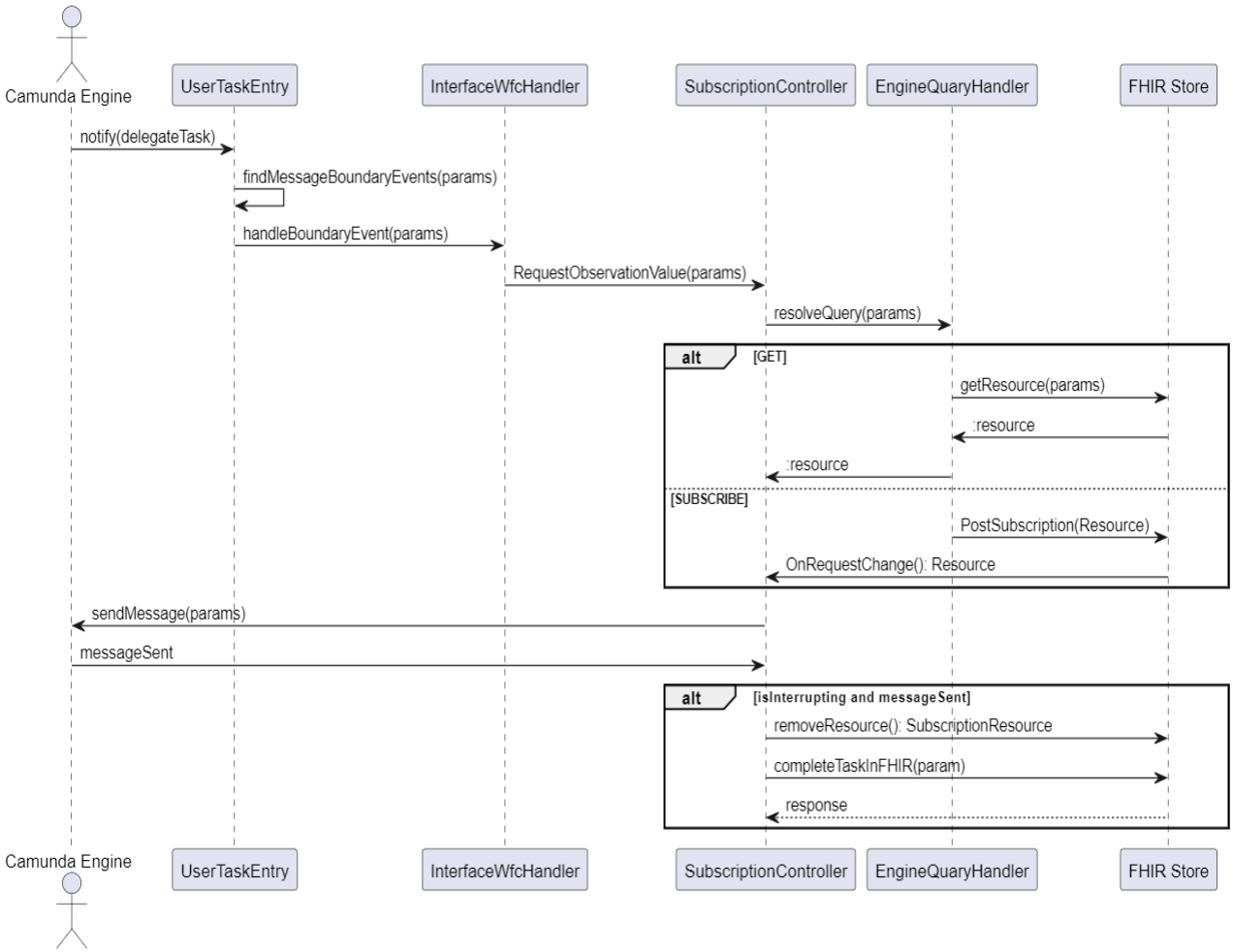

Figure 4.13 Sequence diagram data exchange within HSWC

Figure 4.13 depicts a sequence diagram that illustrates the interaction between the BPM engine, WFC and the FHIR store during the process of sending a FHIR query, receiving a response, and completing a task based on that response. The params in this context are to show that there are parameters to pass, but due to many parameters, it was better to use as 'params' and check the methods in the components listed in the source code which is under Philips labs repository[7]. Here is a breakdown of the key components and steps involved in this sequence:

1. **BPM Engine Initiation**: The sequence begins with the BPM engine initiating a FHIR query. This action is typically triggered by a specific event in the workflow, such as the completion of a User Task that requires additional patient data.
2. **Sending FHIR Query**: The BPM engine constructs a FHIR query based on the requirements of the workflow capability. This query is sent to the WFC and after processing the query and depending on the query type (GET, FETCH, and SUBSCRIBE), discussed in 5.2.4. then sent to FHIR server, which is responsible for managing and providing access to healthcare data.
3. **Response Message**: After processing the query, the FHIR server sends a response back to the BPM engine when data is available. This response contains the requested data, which may

[7] philips-labs/workflow-capability (github.com)

include patient information, clinical observations, or other relevant details necessary for the workflow.

4. **Receiving the Message**: The BPM engine receives the response message from the FHIR server. This step is crucial as it allows the engine to access the data needed to make informed decisions about the next steps in the workflow.
5. **Completing the Task**: With the data received from the FHIR server, the WFC sends a request to complete the activity in FHIR store.
6. **Workflow Continuation**: After completing the task, the BPM engine continues to the next step in the workflow, which may involve further processing, additional tasks, or interactions with other systems.

This sequence diagram effectively illustrates the integration of the BPM engine, WFC and the FHIR standard, highlighting how the WFC achieves the control and communication between the BPM engine and the FHIR store. It plays a crucial role in maintaining synchronization of workflow between the execution module which is the workflow engine (BPM engine) and the Data module which is the FHIR store.

5.Implementation

The enhancement and implementation phase of the HSWC involves several critical steps to ensure that the system operates effectively and meets the needs of healthcare providers. This section outlines the key components of the implementation process, including environment configuration, integration with the Pulse Physiology Engine, data mapping, and workflow model development.

5.1 Environment Configuration

The first step in the implementation process is setting up the development environment. The HSWC is designed to run within a Docker container, which ensures consistency across different deployment environments. This containerization facilitates easy deployment and testing of the application.

The HealthSuite Workflow Capability (HSWC) was configured to ensure seamless integration and functionality. Key components include:

- **Pulse Engine-FHIR Interface App**: A simulation tool that provides realistic physiological data for patient management. It is developed in Jupyter notebook.
- **FHIR Server**: A server that supports the FHIR standard for data exchange. It provides API to communicate with different components of HSWC.
- **Workflow Capability Software**: The core software engine that controls the workflow and manages knowledge model. It is designed and developed by Sprint-Boot framework.
- **Workflow Modeler**: Camunda Modeler was used to develop workflow models.
- **Workflow Engine**: The software that manages and executes clinical workflows. In this prototype, Camunda Engine was used to execute the workflow models.
- **Caregiver App**: This application manages patient registration, creating and managing care plans and deploying workflow models.

5.2 Workflow Models

The HSWC allows for the deploying of configurable BPMN workflows and DMN that can adapt to various clinical scenarios. The existing developed clinical workflow models are instrumented through Receive Task attached to Data Object Reference communicate with other components of HSWC. However, in this development, it is not only clinical workflows that are instrumented through User Task surrounded by Message Boundary Events to communicate with the rest of the HSWC components but also improving the workflow models by adding Time Query (TQ) as discussed in 5.2.4.

5.2.1. User Task

User Tasks are key elements in BPMN that represent steps in a process that should be completed by users in the application side. A full mapping of the User Task is presented in the report [1], Table 5.2.

5.2.2. Message Boundary Event

As explained in the report [1], the FHIR standard facilitates the exchange of healthcare information electronically and is widely adopted for its flexibility and interoperability. For clinical workflows to make informed decisions, the Workflow Engine must query the FHIR store to retrieve necessary patient information. This requires a structured approach to data retrieval, effectively managed through BPMN.

When the token in a BPMN model reaches a User Task linked to a Message Boundary Event, a query is automatically dispatched to the WFC. The WFC manages communication between the Workflow Engine and the FHIR Store, sending a request for the specific patient data required for the workflow.

Once the data is retrieved, a RESTful message containing the information is sent back to the BPM engine by the WFC. The Message Boundary Event captures this message, and the data is stored locally within the BPM engine. From this point forward, the data can be referenced and utilized throughout the remaining workflow, ensuring that all subsequent tasks have access to the necessary patient information.

5.2.3. Storing FHIR Queries in Message Boundary Event

To query a specific FHIR Object, it is crucial to create a precise reference to the desired FHIR Object and to determine where to store this reference within the BPMN model. In BPMN, a Message Boundary Event has several fields available for additional data: ID, Name, Documentation field, global message reference, and message name. Among these, the Documentation field is the most suitable for storing a custom query because it allows for flexible interpretation.

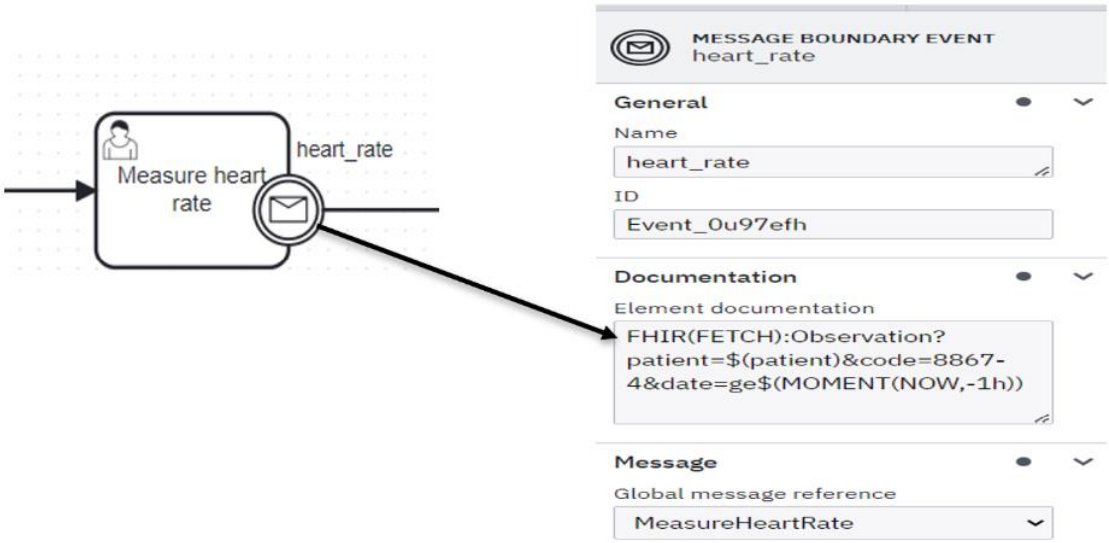

Figure 5.1 Data representation in Message Boundary Event

5.2.4. Constructing FHIR Queries

To standardize the query process, we have adopted a structured FHIR Query format that comprises five essential sub-parts, as illustrated in Figure 5.1: Data Type, CRUD Operation, FHIR Object, FHIR Object Query, and Time query. Each sub-part serves a distinct purpose in defining the query, and their inclusion or exclusion can significantly influence the workflow's functionality and flexibility.

The report [1], the foundation for this project, explained most of the parts. Below, we explore newly added and improved sub-parts in detail and discuss their implications.

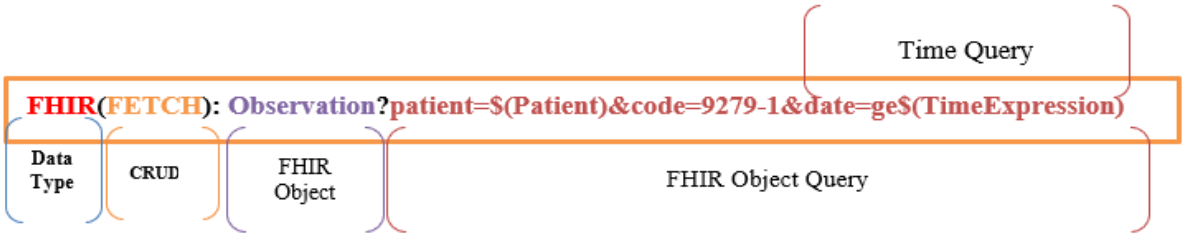

Figure 5.2 FHIR Query structure

Data Type:
The Dara Type specifies the type of object to retrieve, ensuring future flexibility for various data types.

CRUD Operation:
The CRUD Operation sub-part determines the type of action the workflow should take on the FHIR data. In our workflow, the most used CRUD operations are:

- **GET**: Retrieve the current value of FHIR Object.
- **FETCH**: Retrieve the current value of FHIR Object if available; otherwise, wait until the data becomes available. This operation includes both GET and SUBSCRIBE.
- **SUBSCRIBE**: Subscribe to updates for FHIR Object, receiving data as soon as it changes.

These operations allow the workflow to interact dynamically with the FHIR Store, either by retrieving existing data or by setting up mechanisms to monitor future changes. If the CRUD Operation sub-part is omitted, the system defaults to the GET operation, which is sufficient for most data retrieval scenarios. However, omitting this sub-part limits the flexibility of the workflow, particularly in cases where the workflow might need to update or create new data in the FHIR Store.

FHIR Object
The FHIR Object identifies the specific FHIR resource needed, enhancing versatility beyond just FHIR Observations.

FHIR Object Query
The FHIR Object Query identifies exact data within a FHIR resource using parameters, crucial for retrieving relevant information and avoiding broad queries.

Time Query: Filtering Data by Time
The **Time Query** (referred to as Time Expression in Figure 5.2) is a sub-part of FHIR Object Query that allows the workflow to filter data based on specific time. In clinical settings, the timing of observations or events is often crucial for making accurate decisions. The Time Query enables the workflow to specify a time range for the data being retrieved, ensuring that only relevant and timely information is returned. Two primary expressions are used within the Time Query:
1. NOW Expression: This expression retrieves data that is current or very recent. It is particularly useful for workflows that require the latest information, such as the most recent vital signs or lab results. Figure 5.3 provides an example of query representation with keyword NOW in the documentation part of the Message Boundary Event.

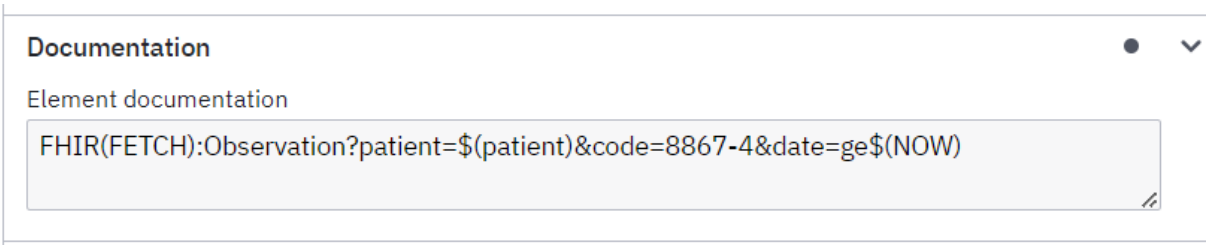

Figure 5.3 Query structure using keyword NOW

This query fetches the observation for the specified patient and code, where the observation date is greater than or equal to the current time.

2. MOMENT Expression: The MOMENT expression allows for querying data relative to a specific point in time. This can be used to retrieve data from a particular time window, such as all observations within the last 1 hour. Figure 5.4 shows a clear representation with keyword MOMENT and variable $var inside the MOMENT expression.

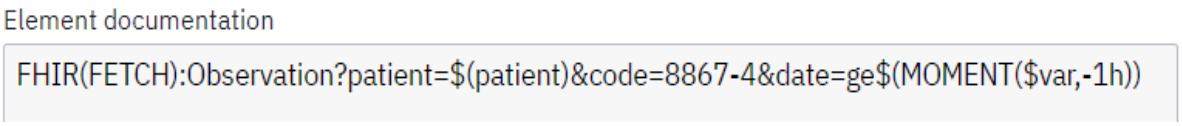

Figure 5.4 Query structure using keyword MOMENT and arbitrary date variable

In this case, the query retrieves observations where the observation date is greater than or equal to a moment defined by a variable adjusted by a time offset (e.g., one hour before the variable's value).
The “$var” in Figure 5.4 shows the specific scenario of certain date value, for instance the time by which some certain observation has seen. So, by expressing it, one can retrieve the heart rate that is seen 1 hour after some FHIR resource observed date.

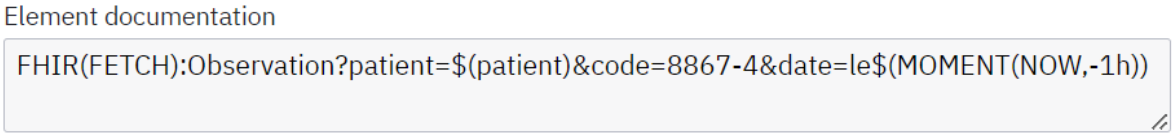

Figure 5.5 Query structure using keyword MOMENT and NOW as a keyword

Figure 5.5 shows the replacement of “$var” with NOW, which can clarify the meaning of the expression. It retrieves data that has been seen 1 hour before now.

In FHIR (Fast Healthcare Interoperability Resources), the terms "le," "ge," "gt," and "lt" are used as comparison operators in search queries. They are part of the FHIR search syntax that allows clients to filter resources based on specific criteria. Here is what each operator means:

- **le**: Less than or equal to. This operator is used to find resources where a specified field's value is less than or equal to a given value.
- **ge**: Greater than or equal to. This operator is used to find resources where a specified field's value is greater than or equal to a given value.
- **lt**: Less than. This operator is used to find resources where a specified field's value is less than a given value.
- **gt**: Greater than. This operator is used to find resources where a specified field's value is greater than a given value.

These operators can be used in conjunction with various FHIR resource attributes to refine search results.
The inclusion of the Time Query ensures that the workflow retrieves data that is not only accurate but also timely, which is crucial for clinical decision-making. Omitting this sub-part would limit the workflow's ability to filter data by time, potentially leading to the use of outdated or irrelevant information.

5.3 Integration Pulse Physiology Engine and HSWC

The Pulse Engine repository contains all the code needed to build the engine. Pulse is written in C++ and provides an interface for other languages, including Java, C, C#, and Python. The integration of the Pulse Physiology Engine into the Workflow Capability is achieved through its libraries provided by Pulse Physiology Engine developed in Jupyter notebook. Python application called Pulse Engine – FHIR Interface App was developed to integrate the Pulse Physiology Engine with HSWC particularly with HL7 FHIR.

5.3.1. Pulse Engine-FHIR Interface App

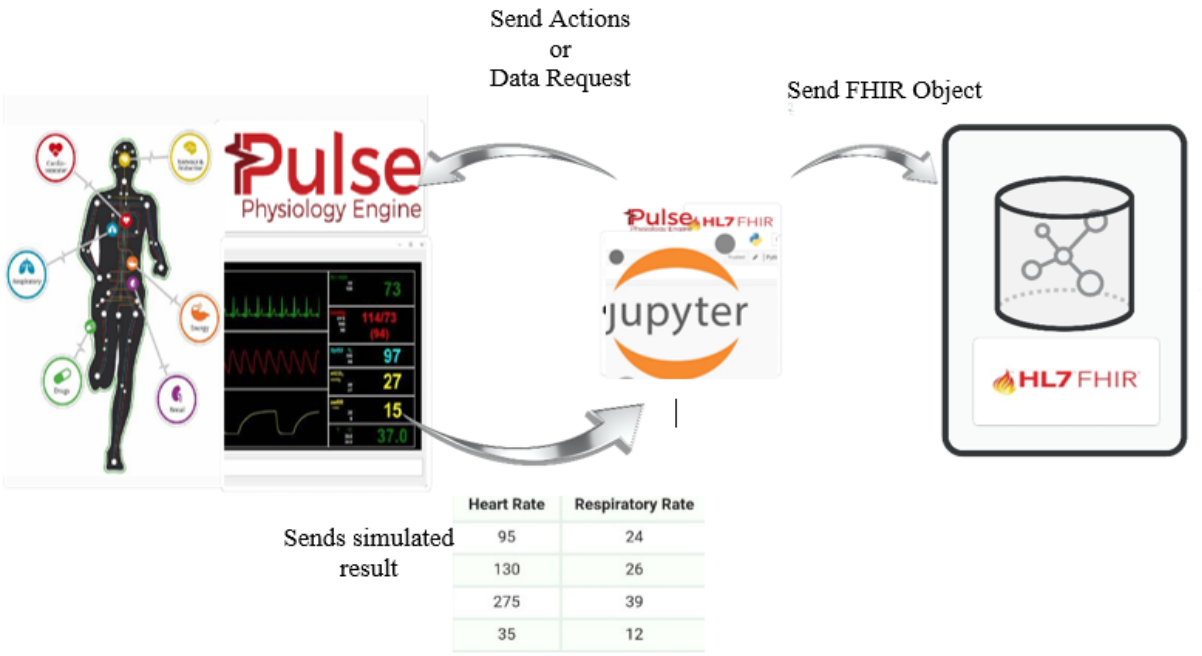

Figure 5.6 Implementation of Pulse Engine-FHIR Interface App

Figure 5.6 illustrates the implementation of the Pulse Engine-FHIR Interface App, which serves as a bridge between the Pulse Physiology Engine and the FHIR (Fast Healthcare Interoperability Resources) store. This interface is crucial for integrating simulated physiological data into the HSWC system.

The implementation is done using a Jupyter notebook, which provides an interactive environment for developing and testing the integration. The notebook includes buttons to simulate physiological actions, specifically "Block Airway" and "Unblock Airway". These buttons trigger 20-second simulations of airway blockage and unblockage respectively. The simulation is stable 20-30 seconds after action is taken [11]. The Pulse Engine generates simulated physiological data, including heart rate and respiratory rate, based on the triggered events. The generated data is converted into FHIR-compliant format. This likely involves mapping the Pulse Engine's data structures to corresponding FHIR resources (e.g., Observation). The converted data is then sent to the FHIR store, making it available for use by other

components of the HSWC system. The notebook also includes functionality to visualize the simulated data, likely in the form of graphs or charts.

Design decision: *Single observation value (i.e. single observation value of heart rate and respiratory rate) is sent to the FHIR store in each simulation action because the instrumented clinical workflow models need only a single Observation value from FHIR store at a time. The other simulation values are stored in CSV file for further analysis.*

In summary, the Pulse Engine -FHIR Interface App demonstrates the communication between the FHIR store and the Virtual Patient. The Pulse Engine-FHIR Interface App sends the heart rate and respiratory rate observation to FHIR store from the Virtual Patient (Pulse Physiology Engine).

5.4 Closed-Loop Scenario

In this project, we introduce the concept of a closed-loop system. This system involves a FHIR store, which collects data such as heart rate and respiratory rate from a virtual patient. The virtual patient, a simulated representation of a real patient, receives commands or medication information from the FHIR store based on specific decision-making in the clinical workflow.

5.4.1. Closed-Loop Design

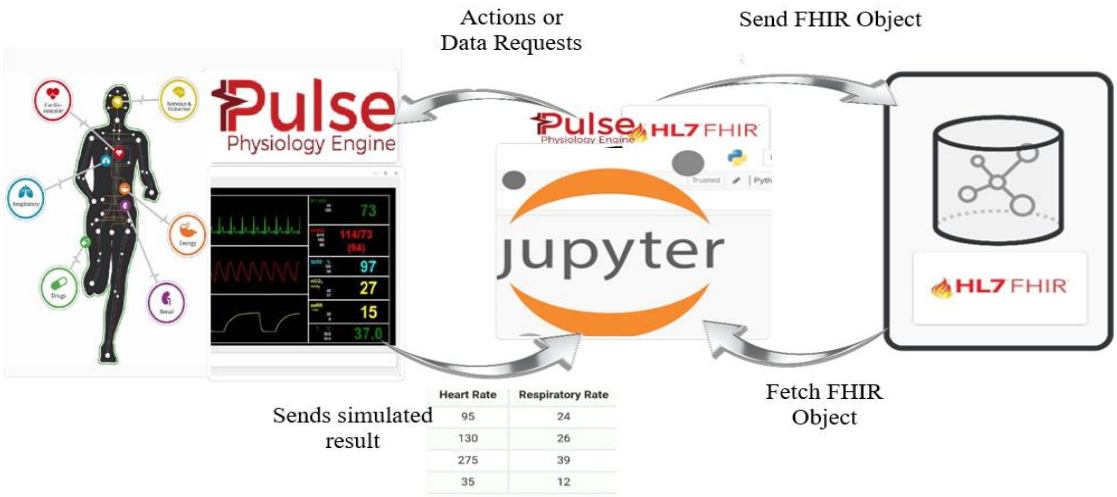

Figure 5.7 Implementation of closed-loop scenario

Figure 5.7 illustrates the implementation of a closed-loop system utilizing the Pulse Engine-FHIR Interface App. This application fetches FHIR objects, such as observations, interventions, or medications, from the FHIR store. Based on the retrieved data, the Pulse Engine-FHIR Interface App sends instructions to the Pulse Physiology Engine to simulate changes in the patient's physiological state.

To simulate the closed-loop system, we developed sample clinical workflow models using BPMN, incorporating Message Boundary Events as shown in Figure 5.8 and Figure 5.9. These models simulate scenarios where the FHIR store receives respiratory rate data from the virtual patient and sends it to a BPM engine (e.g., Camunda). A DMN business rule within the BPMN model is then used to calculate

the respiratory rate based on the received data[8]. The full BOMN models can be found Philips labs repository[9].

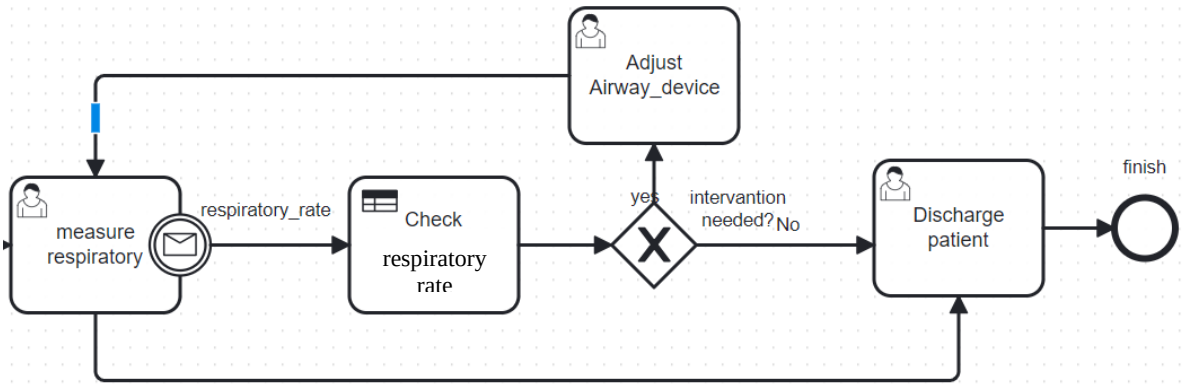

Figure 5.8: Sample input model for closed loop using interruptive Message Boundary Event

As shown in Figure 5.8, the instrumented clinical workflow model is constructed using interruptive boundary message event.

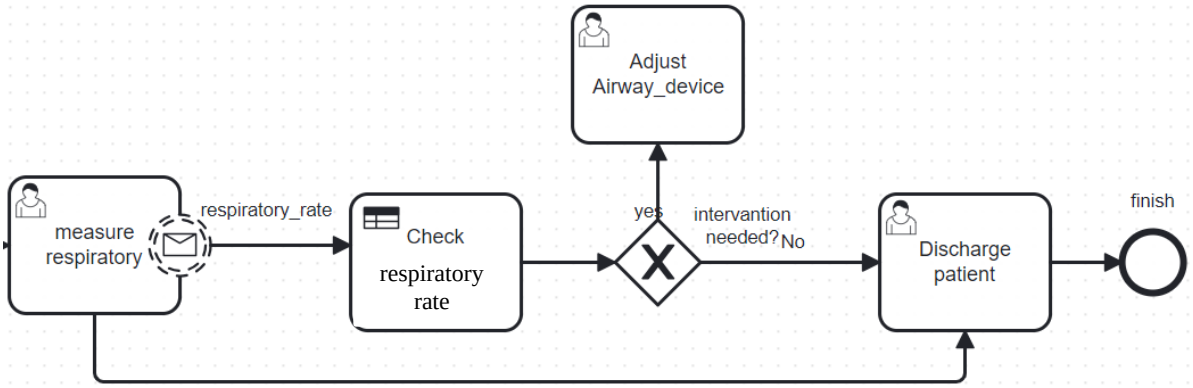

Figure 5.9: Sample input model for closed loop using non-interruptive Message Boundary Event

As shown in Figure 5.9, the instrumented clinical workflow model is constructed using non-interruptive boundary message event.

[8] https://github.com/philips-labs/workflow-capability/tree/feature/pulse-physiology_FHIR
In Philips labs repository, Features branch contains detailed information and experiment on closed loop
[9]https://github.com/philips-labs/workflow-capability/tree/feature/workflow_models/sepsis/v3/BPMN/CLOSED-LOOP

```python
data_management = DataManagement(patient_id, config_file)
simulation_core = SimulationCore(patient_id, data_management)
look_back = Look_back(patient_id, config_file, simulation_core)

start_time = time.time()
try:
    while time.time() - start_time < 300:  # Run the loop for 5 minutes
        instruction = look_back.fetch_instruction("232209000")
        if instruction:
            observation_time = datetime.fromisoformat(instruction['effectiveDateTime'].replace('Z', '+00:00'))
            current_time = datetime.now(pytz.utc)
            # Check if the observation is within the last 1 minute
            if (current_time - observation_time).total_seconds() <= 60:
                print(f"Observation fetched at {datetime.now()}:")
                print(instruction)
                look_back.handle_instruction(instruction)
                draw()
            else:
                print(f"Instruction is older than 1 minutes and will be ignored. Looking for another instruction")
        time.sleep(30)
except KeyboardInterrupt:
    print("Fetching data stopped.")
finally:
    print("Simulation completed.")
```

Figure 5.10: Loop-back to receive Instruction from FHIR store

As shown Figure 5.10, the closed-loop scenario operates without relying on manual input through buttons. Instead, the simulation actions and instructions are automatically generated by the Pulse Engine-FHIR Interface App based on the data fetched from the FHIR observations.

5.4.2. Result and Analysis

```
209     main()
[11]
...  Enter the patient ID:  180
     Initial state of the patient:
     ------------------------------
     Heart Rate value: 85.714286
     Respiratory Rate value: 15.957447
     No new observations found for patient 180 with code 232209000.
     No new observations found for patient 180 with code 232209000.
     Observation fetched at 2024-08-27 20:30:02.539044:
     {'resourceType': 'Observation', 'id': '190', 'meta': {'versionId': '1', 'lastUpdated': '2024-08-27T20:29:18.28
     Obstructing patient airway...
     Heart Rate value: 88.235294
     Respiratory Rate value: 0
```

Figure 5.11: Reading instruction to block airway from FHIR store

For this scenario, a sample nasal airway obstruction code ("232209000") was utilized[10]. When the Pulse Engine-FHIR Interface App is executed and the patient ID is entered, the initial state of the patient is displayed, as shown in Figure 5.11 and sent to FHIR store. When the value for the nasal airway obstruction code is observed in the FHIR store, the "look_back" method in the Pulse Engine-FHIR Interface App retrieves the value and sends an action to simulate airway obstruction in the Pulse Physiology Engine. This action can either block or unblock the airway, as illustrated in Figure 5.11 and Figure 5.12 respectively. The simulation generates corresponding physiological responses, which are then sent back to the Pulse Engine-FHIR Interface App based on the data request. In this simulation, a value of 1 observed in the FHIR store indicates a complete blockage of the airway, while a value of 0 signifies that the airway is unblocked. Consequently, the Pulse Engine simulates the patient's condition based on

[10] https://github.com/philips-labs/workflow-capability/blob/feature/pulse-physiology_FHIR/ppe_fhir.ipynb describes detailed about the purpose of the experiment and the examples used.

the value fetched from the FHIR store and returns the results to the application, as shown in Figure 5.11 and Figure 5.12.

```
Observation fetched at 2024-08-27 20:31:06.919432:
{'resourceType': 'Observation', 'id': '195', 'meta': {'versionId': '1', 'lastUpdated': '202
Obstructing patient airway...
Heart Rate value: 85.714286
Respiratory Rate value: 19.867550
```

Figure 5.12: Reading instruction to unblock airway from FHIR store

The application subsequently maps the simulated data into FHIR format and stores it in the FHIR store for further use by other components of the HSWC system. This closed-loop functionality facilitates continuous interaction between the Pulse Engine-FHIR Interface App and the Pulse Physiology Engine, enabling real-time simulation and monitoring of patient conditions based on data from the FHIR store.

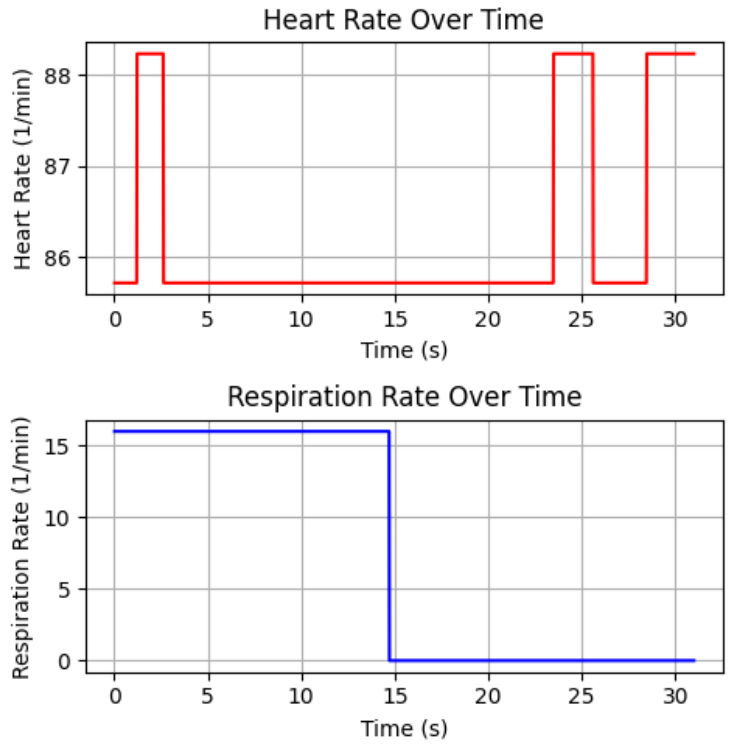

Figure 5.13: Heart rate and respiratory rate after blocking airway for 30 seconds

Figure 5.13 displays the heart rate and respiratory rate after blocking the airway for 30 seconds. The respiratory rate decreases and records a value of zero, indicating that the airway is blocked.

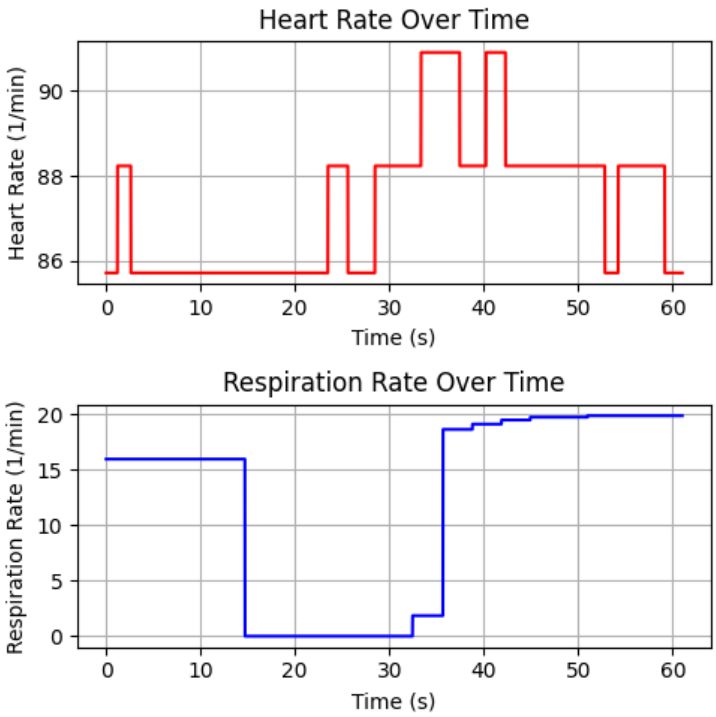

Figure 5.14: Heart rate and respiratory rate after unblocking airway for 30 seconds

Figure 5.14 shows the heart rate and respiratory rate after unblocking the airway for 30 seconds, with the respiratory rate changing from zero to 19.86 breaths per minute.

The goal of this scenario is to demonstrate the bidirectional communication between the FHIR store and the virtual patient. It shows how changes in respiratory rate prompt actions within the clinical workflow, which then either block or unblock the airway obstruction in the virtual patient. While not the most ideal example, it effectively highlights the system's capabilities and direction.

In summary, the closed-loop scenario demonstrates the two-way communication between the FHIR store and the Virtual Patient. The Pulse Engine-FHIR Interface App not only transmits vital signs from the Pulse Physiology Engine but also retrieves instructions from the FHIR store to simulate patient responses based on the resource type and value present in the FHIR store. This capability significantly enhances the simulation of patient conditions.

6. Verification and Validation

The HSWC project employed a comprehensive approach to ensure the system's functionality and reliability through various testing methodologies. These processes included unit testing, integration testing, code reviews, and scenario-based testing, which are categorized under verification and validation (V&V) processes.

6.1 Verification

The verification process involved several key activities to ensure that the system was built correctly according to the specified requirements and design documents.

Unit Testing: Conducted on key components such as the FHIR Query Processing Checker to validate their functionality. A total of 10 test cases were executed, comprising 5 successful and 5 failed scenarios as shown in Figure 6.1.

```
testcase0 = "FHIR(FETCH):Observation?patient=$(patient)&code=11331-6";
testcase1 = "FHIR(FETCH):Observation?patient=$(patient)&code=11331-6&date=le$(MOMENT(NOW,-1h))";
testcase2 = "FHIR(FETCH):Observation?patient=$(patient)&code=11331-6&date=le$(MOMENT($var,-1d))";
testcase3 = "FHIR(FETCH):Observation?patient=$(patient)&code=11331-6&date=le$(MOMENT($var2,1m))";
testcase4 = "FHIR(FETCH):Observation?patient=$(patient)&code=11331-6&date=le$(NOW)";
testcase5 = "FHIR(FETCH):Observation?patient=$(patient)&code=11331-6&date=le$(Kodffjh)";
testcase6 = "FHIR(FETCH):Observation?patient=$(patient)&code=11331-6&date=le$(NEW)";
testcase7 = "FHIR(FETCH):Observation?patient=$(patient)&code=11331-6&date=le$(MOMENT($var,-1g))";
testcase8 = "FHIR(FETCH):Observation?patient=$(patient)&code=11331-6&date=le$(MOMIENT($var2,-1d))";
testcase9 = "FHIR(FETCH):Observation?patient=$(patient)&code=11331-6&date=le$(MOMENT($var3,-1d))";
patient = "Patient/2";
var = "2024-08-25T17:00:00Z";
var2 = "2024-08-25";
var3 = "2024-08-25T17:00:00Zabc";
```

Figure 6.1: Test cases

Based on the test cases given and the variables along with the values to be replaced, the query process function was tested in an isolated environment to verify its ability to construct valid FHIR queries. By replacing process variables and expressions with specific values, the unit tests successfully validated the system's functionality. As shown in Figure 6.2, Test case 0 successfully constructs a query without any date expression. Test cases 1-4 demonstrated the correct use of the MOMENT and NOW expressions for various date-based filtering scenarios. These results confirm the system's capability to generate WFC standard FHIR queries with different date filtering options.

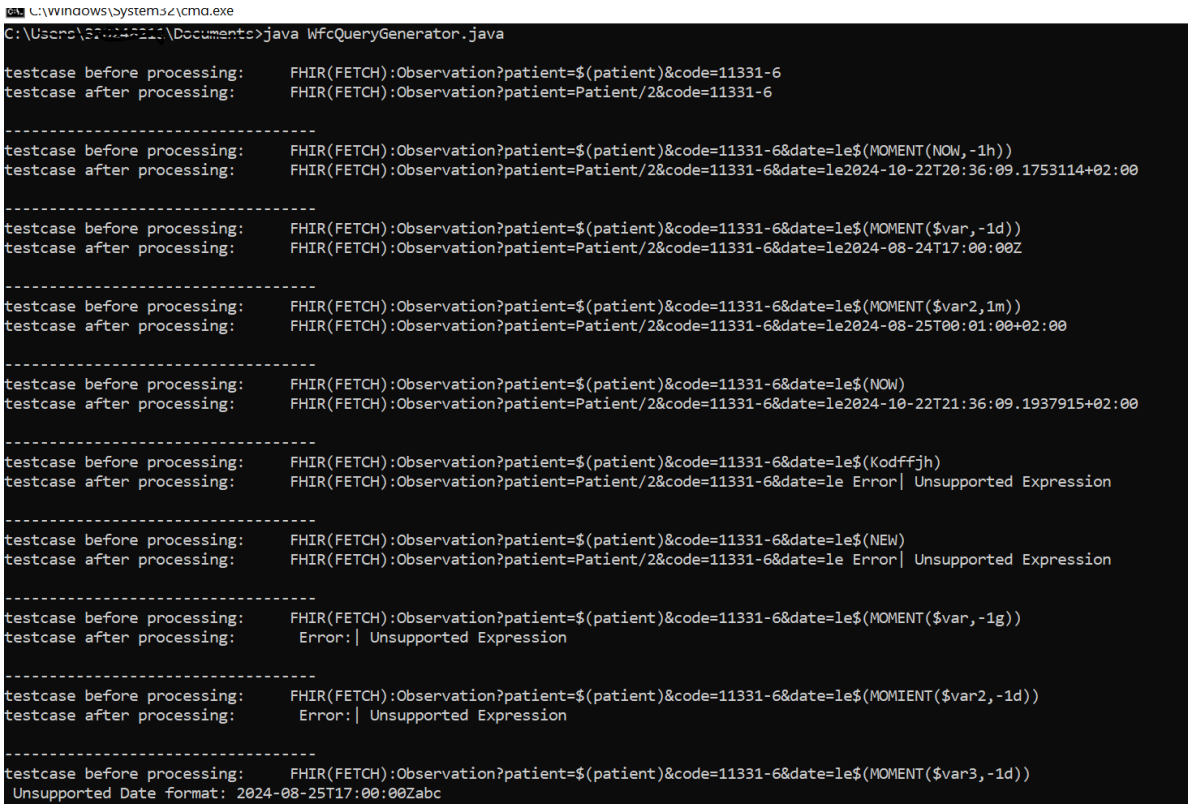

Figure 6.2: Query process function test in isolated environment test result

As shown in Figure 6.2 the unit test also identified several failed cases due to invalid expressions and incorrect date formats. In test case 5, the expression "Kodffjh" was unrecognized, leading to failure. Similarly, test case 6 used the invalid expression "NEW." Test case 7 employed an incorrect flag "g" in the MOMENT function, resulting in a failed query. In test case 8, the misspelling of "MOMIENT" caused the test to fail. Finally, test case 9 used an invalid date format in the "var3" variable, leading to an error. These failed test cases highlight the importance of using valid expressions, correct flags, and proper date formats in FHIR queries.

The unit testing process has successfully validated the functionality of the FHIR Query Processing. The HSWC can handle a variety of date-based filtering expressions and generate valid FHIR queries. Furthermore, it effectively detects and reports errors for invalid inputs, ensuring the integrity of the generated queries.

Code Reviews: Code reviews were performed regularly to maintain coding standards and detect potential issues early in the development cycle.

Integration Testing: Integration testing was carried out between different components of the HSWC at every phase during the enhancement or development of the system. This testing ensured that each component was functionally correct. For example, during modification of Camunda Engine, tests primarily focused on the communication between Camunda and the workflow capability. Additionally, when enhancing or modifying the workflow capability, testing involved the caregiver app, the Camunda engine, and the FHIR store to verify the functionality of subscription services, task auto-completion, and communication with the Camunda engine, such as message sending. Furthermore, integration testing was conducted between the Pulse Engine-FHIR Interface App and the FHIR store to ensure accurate data storage.

Scenario-Based Testing: Scenario-based testing included test cases that verified the system's behavior with both existing and new models, heart rate data retrieval using time filters, and virtual patient integration. All test cases were tested on Sepsis protocol use case[11].

- **Test Case 1: Workflow Model Based on Receive Task**
  This test verified the system's ability to process existing models correctly. The same behavior was observed as in the old model after deploying the Receive Task-based model and entering value greater than the threshold for the task "Measure Sobriety". After completing the task on the Caregiver App, the workflow continued as expected. Thus, the test confirmed that the system works for the old models configured based on the Receive Task with Data Object Reference.

- **Test Case 2: Workflow Model Based on Message Boundary Event**
  This test aimed to check if the system functions correctly with newly developed workflow models using Message Boundary Events. The BPMN model with the Message Boundary Event was deployed, and the system was run with the previously entered value from Test Case 1 to verify if the system could retrieve the value and automatically change the task status to completed in the FHIR store. Consequently, the system retrieved the value, completed the task in FHIR, and the workflow continued in the Camunda engine. Thus, we verified that the system is functioning as intended for the newly developed models based on the objectives of the research.

- **Test Case 3: Workflow Model Based on Message Boundary Event Configured with "NOW" Expression**
  For this test, we configured the FHIR Query format as "FHIR(FETCH):Observation?patient=(patient)&code=11331-6&date=le$(NOW)" in the BPMN model to retrieve data from the FHIR store, specifically to access observations not older than the current time. This test was based on the previous test case, which involved existing observations recorded before five minutes ago. The system did not fetch the data initially; however, after entering a value less than the threshold for the "Measure Sobriety" task from the Caregiver App, the workflow capability notified that a new observation for sobriety had been recorded. Subsequently, the workflow capability sent a message to the Camunda engine, allowing the workflow to continue to the next tasks: measuring heart rate, measuring respiratory rate, and measuring temperature. The workflow capability then completed the task in the FHIR store. Thus, we confirmed and verified that the system successfully works with the FHIR Time Query Expression.

- **Test Case 4: Virtual Patient Integration**
  In this test case, a virtual patient was integrated to obtain realistic patient vital signs, specifically heart rate and respiratory rate. Building on Test Case 3, the tasks to measure heart rate and respiratory rate were expected to fetch values from the FHIR store. After running the Pulse Engine-FHIR Interface App and entering the patient ID, the application requested values from physiological model software and the simulated values for heart rate and respiratory rate were displayed as 85.71 and 15.95, respectively. Observations of the heart rate and respiratory rate were recorded in the FHIR store. Subsequently, the system triggered the workflow capability, and both tasks were completed in the FHIR store. The Camunda engine also indicated that both tasks had moved to the next step. This test case demonstrated that the system successfully simulates the virtual patient and verifies the integration effectively.

[11] https://github.com/philips-labs/workflow-capability/tree/main/workflow_models/sepsis/v3

6.2 Validation

The validation process focused on ensuring that the system met user needs and expectations through various testing techniques.

E2E testing was a critical validation mechanism that assessed the entire system's functionality through scenario-based tests. These tests included verifying the system's ability to process workflow models based on a sepsis protocol, evaluating system behavior with new models, testing heart rate data retrieval using time filters, and integrating a virtual patient with the HSWC.

Acceptance testing was another important validation step, where the system was deployed to the main repository as open-source software and tested according to a predefined test plan. Stakeholders reviewed, demonstrated, and accepted the system thorough validation.

Structured version control and stakeholder engagement were integral to the validation process. Code and documentation were reviewed prior to acceptance, and pull requests were systematically reviewed by supervisors before merging into the development branch. The final review by stakeholders led to the merging of the code into the main branch for public release. Docker was utilized to containerize most system components, ensuring consistency across different environments.

In conclusion, the V&V process conducted in this project has been instrumental in ensuring the reliability, functionality, and effectiveness of the HSWC. The comprehensive approach to V&V involved multiple testing methodologies tailored to the unique challenges presented by the system's architecture and component dependencies. As shown in the verification step, integration testing and E2E testing were commonly used in this project. This choice was driven by the high interdependence of the application components and the presence of open-source elements, which made it suitable compared to other tests. To address this limitation, we opted for integration testing and model-based or scenario-based E2E testing, allowing for a thorough assessment of how well the components work together and how the system performs in real-world scenarios. The V&V activities confirmed that the HSWC meets the specified requirements and effectively optimized clinical workflows, demonstrating that the system enhances operational efficiency and improves patient care by integrating Virtual Patient.

Furthermore, the involvement of stakeholders throughout the V&V process was critical, as their insights and feedback guided the testing efforts and ensured that the system aligns with user expectations.

7. Results and Conclusions

7.1 Results

Various workflow modeling approaches were explored and evaluated using a Pugh matrix. The "Message Boundary Events on User Task" approach was identified as the most advantageous, improving process flow clarity, optimizing workflow model creation, and facilitating data-driven flow continuation.

The Pulse Physiology Engine was successfully integrated with the HSWC to enable realistic simulation of patient physiological responses. Running the simulation on jupyter notebook based on the provided buttons to block airway and unblock for 20 seconds to block the airway and 20 seconds to unblock the airway and sent to FHIR store demonstrated the effectiveness of this integration.

A structured FHIR Query WFC standard (readable by WFC) format was adopted, containing five essential sub-parts: Data Type, CRUD Operation, FHIR Object, FHIR Object Query, and Time Query. This standardized approach enhanced the workflow's functionality and flexibility in querying the FHIR Store.

Comprehensive verification and validation methods were employed. These included unit testing, code reviews, and scenario-based validation with five workflow model scenarios, which were part of the E2E Integration Testing conducted to validate the correct functioning of the HSWC. The test report confirms that all five test cases passed successfully, indicating that the system's components are functionally working as intended, despite some tolerable errors encountered due to the use of open-source components such as Camunda and HAPI FHIR. Additionally, a structured development process with version control using Git branches was followed throughout the project.

7.2 Conclusions

The goal of this project is to explore and identify an alternative solution for the existing workflow that reduces the number of elements in clinical workflow models, enables the automatic closure of activities based on available data, and integrates Virtual Patient.

The project was driven by three primary research questions:
RQ1: *How can different instrumented clinical workflow modeling methodologies be explored and evaluated to identify the most effective alternative method that reduces the number of elements while maintaining functionality?*

To address this question, three workflow modelling approaches were compared using a Pugh matrix, with the existing workflow model serving as the base. The comparison identified the "Message Boundary Event attached to User Task" approach as the most effective, reducing complexity while maintaining the necessary functionality.

RQ2: *Can the WFC automatically close activity (User Task) in the FHIR store when the required data becomes available, and what specific criteria must be established for this functionality to operate successfully?*

This question was addressed by designing a **FHIR Query** and **FHIR Time Query** within the Message Boundary Event implementation. Enhancements to the Workflow Capability (WFC), as described in Section 4.4 and illustrated in Figure 4.13(sequence diagram), demonstrated how a task is automatically completed when relevant data becomes available.

RQ3: *How can the physiological model software be integrated with the HSWC to simulate realistic patient vital signs to test clinical workflows?*

To answer this question, the **Pulse Engine-FHIR Interface App** was designed and implemented. This app integrates the **Pulse Physiology Engine** with HSWC, particularly with the FHIR store component, as discussed in Section 4.2.4. The system effectively simulated realistic patient vital signs, such as heart rate and respiratory rate, and integrated them into clinical workflows for testing purposes.

Overall, the project successfully enhanced the HSWC by integrating physiological model software, optimizing workflow models, and enabling the automatic closure of activities based on FHIR data availability. An Agile methodology, involving regular communication, stakeholder engagement, and phased delivery, was instrumental in achieving the project goals.

7.3 Recommendations and future work

- Further optimization of workflow modeling approach could be explored to enhance the efficiency of clinical workflow processing, for example other BPMN Elements like Service Task, Script Task.
- Further investigations on representing the FHIR Query expression in the workflow models in addition to the NOW and MOMENT expressions for flexible data fetching from FHIR store.
- Expanding integration of Virtual Patient with the HSWC, particularly the closed loop scenario could enable more comprehensive testing and validation of clinical workflows by adding more FHIR resources like Medication Request.
- Collaborating with the open-source community to address and resolve the encountered errors in Camunda and HAPI FHIR could contribute to the overall stability and reliability of these components.
- Investigating the potential integration of machine learning algorithms with HSWC could enable more intelligent and adaptive clinical decision support.

7.4 Lessons Learned

I learned a lot of valuable lessons throughout this project. One of the biggest was the importance of getting early and regular feedback from stakeholders. Checking in frequently helped keep the project on track and made sure we were always aligned with our goals. It allowed us to make small adjustments along the way, which prevented bigger problems from popping up later.

Clear communication was another essential part of our success. I realized that keeping open, consistent communication between the team and stakeholders helped avoid misunderstandings and made everything run more smoothly. Regular meetings, well-documented updates, and being transparent about decisions really made a difference.

Risk management also stood out as a crucial practice. By planning for risks right from the start, we were able to spot potential issues early and come up with ways to manage them before they could turn into bigger challenges. Not only working with Agile methodology but also splitting the project in phases (slices) was also a game-changer for me.

On the technical side, I had the chance to broaden my knowledge in a few areas. I deepened my understanding of BPMN and explored tools that help visualize complex workflows, which was incredibly helpful. I also got introduced to HL7 FHIR, which was totally new to me and expanded my understanding of healthcare data standards—a critical aspect in today's healthcare IT. Additionally, I gained insights into human physiology and simulations, and seeing how these are used in real-world healthcare was eye-opening.

From a personal growth perspective, this project provided a huge opportunity for development. I improved my ability to manage people and engage with stakeholders, learning to communicate effectively with different groups, manage expectations, and leverage timely feedback to drive success. I enhanced my skills in requesting feedback and in identifying and prioritizing the most important aspects. Above all, I developed my ability to determine the appropriate level of detail, knowing when to be specific and when to stay general. These soft skills, alongside the technical ones, were key personal takeaways from this project.

8.Stakeholder Analysis

Effective stakeholder engagement is crucial for the success of the Philips' HealthSuite Workflow Capability (HSWC) project. The stakeholders can be categorized into direct and indirect participants, each with distinct roles, expectations, and concerns.

Stakeholders

Table 8.1 provides a clear overview of the stakeholders from Philips HPM Team and Eindhoven University of Technology, highlighting their specific roles and the concerns they bring to the project.

Table 8.1 direct stake holders of the project

Stakeholder Group	Stakeholder Name	Role Description	Primary Concerns
Philips	Bas Bergevoet	Serve as project supervisor within Philips, acting as the primary interface for all project demands.	Successful integration of the physiological model software with the Workflow Capability; continuous enhancement of the Workflow Capability to meet the evolving needs in patient monitoring.
	Patrick Bonné	Serve as project mentor within Philips acting as the primary interface for all project demands.	Successful integration of the physiological model software with the Workflow Capability; continuous enhancement of the Workflow Capability to meet the evolving needs in patient monitoring.
	Maessen Ralph	Serves as Product owner, recommends tools to integrate to the project.	Successful integration of the physiological model software with the Workflow Capability.
Eindhoven University of Technology	Renata Medeiros de Carvalho	Project supervisor from the TU/e, responsible for overseeing the architectural development and documentation of the project.	Ensuring the successful delivery of the project and maintaining high standards for the final project report.
	Yanja Dajsuren	Program director of Engineering Doctorate Software Technology (EngD ST) program.	The overall quality of the project outcomes, sustaining future collaboration with Philips, and the successful graduation of all EngD ST trainees.

9.Project Management

9.1 Methodology

The HSWC project adopted an Agile methodology to facilitate flexibility, iterative development, and continuous improvement throughout the project lifecycle.

Agile Methodology

Sprint Structure: The project was organized into two-week sprints, allowing the team to focus on specific tasks and deliverables within manageable timeframes. Each sprint began with a planning session to prioritize tasks.

1. **Weekly Stand-up**: to ensure ongoing communication and quick resolution of impediments, weekly stand-up meetings were held every Monday. These brief meetings provided me with the opportunity to share progress, discuss challenges, and align on goals.
2. **Sprint planning:** each sprint was started by sprint planning with tasks defined for the two weeks sprint, usually done every second week of Wednesday.
3. **Sprint review:** sprint review was done to review the tasks and at the end of sprint.

Iterative Development: The Agile methodology enabled the team to adopt an iterative approach to development. By breaking down the project into smaller, manageable phases called “slices”.

Tools used

The project initially began using Jira for project management; however, due to accessibility issues, it was transitioned to Microsoft Planner within Teams, as illustrated in Figure 9.1. Microsoft Teams also served as the primary communication channel, allowing me to manage all project files, sprints, and calendars within a single application.

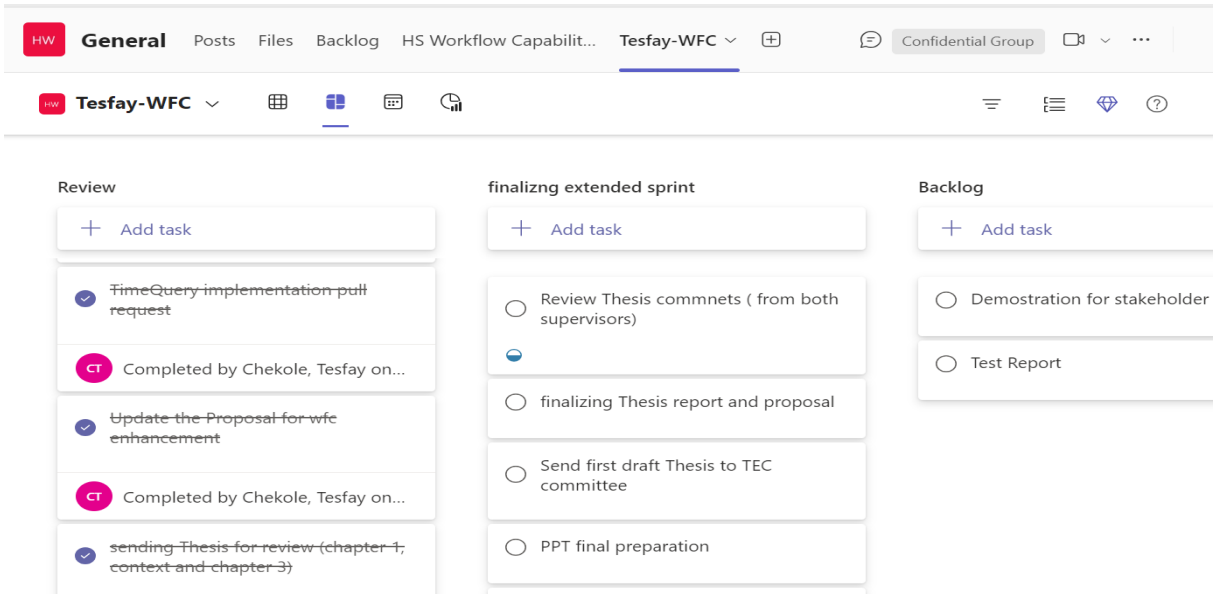

Figure 9.1 Screenshot of the of last sprint on Microsoft Planner tool

The project was divided into seven major phases, each characterized as a "slice."

Each slice culminated in an end-to-end component demo showcase, demonstrating tangible progress and functionality to stakeholders.

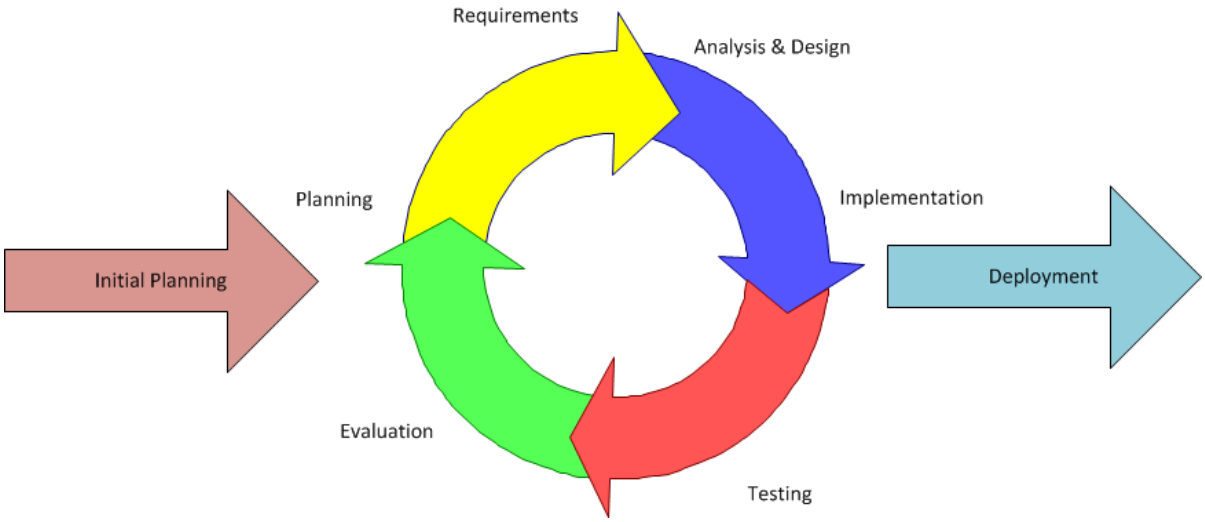

Figure 9.2 Agile methodology

After end-to-end demonstrations, the code has been pushed and considered as deployment, as shown in Figure 9.2. Each deployment was taken after demonstrations. This iterative process was particularly beneficial for this project with multiple components.

For the deployment process and code collaboration, we use GitHub, as the codebase repository was stored in GitHub, under Philips Labs.

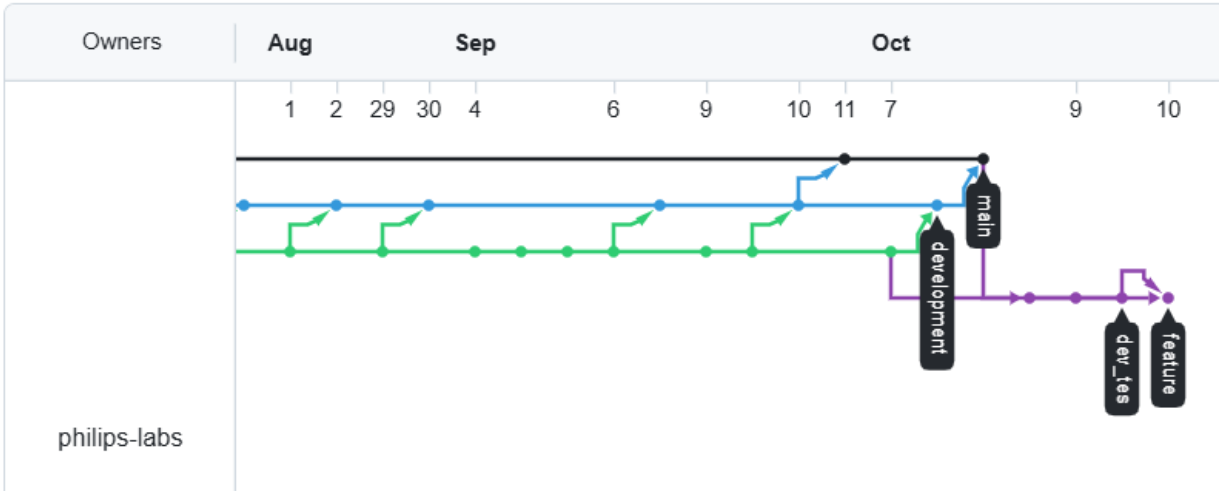

Figure 9.3: GitHub network graph

Figure 9.3, describes a timeline of commits to a code repository. Different branches (main, development, feature, dev-tes) are visible, with commits represented by vertical lines. The arrows show how changes flow between branches[12].

9.2 Communication and Decision making

Regular and structured meetings ensure continuous alignment and adaptability to changes:

[12] https://github.com/philips-labs/workflow-capability/network

- **Weekly Stand-up:** on Mondays help to stay aligned on weekly goals and address any immediate issues.
- **Stakeholder Meetings:** on Wednesdays ensure that project expectations are managed, and that feedback is integrated promptly.
- **Bi-weekly sprint review and sprint planning:** these sessions help to assess progress, identify potential roadblocks, and adjust the project plan as needed.
- **TU/e Supervisor Meetings:** on Tuesdays provide academic oversight and ensure the project meets educational standards.
- **Monthly PSG Meetings:** focus on strategic oversight and resource allocation.

9.3 Project timeline

This section explains the timeline of the project. It is structured into seven major phases spanning from January to October. It begins with the **Initialization Phase** in January and February, focusing on project setup and discovery activities, including initial meetings and planning. In March and April, **Phase-1** involves developing mock physiological model software, while **Phase-2** integrates the Virtual Patient (Pulse Physiology Engine) and designs the integration strategy. **Phase-3** in May to July focuses on containerizing applications and setting up the environment. Concurrently, **Phase-4** optimizes workflows and integrates different systems. August and September are dedicated to **Phase-5**, which involves finalizing the phase 4 and closed-loop testing. The project concludes from September and October with the **Closing Phase**, emphasizing system documentation, presentation preparation, and final deliverables. Figure 9.4 shows the project phases and their corresponding activities.

The reason why there are parallel phases, is that because we use review mechanisms in both and documentation, new phases or tasks may be started before finishing the previous phase.

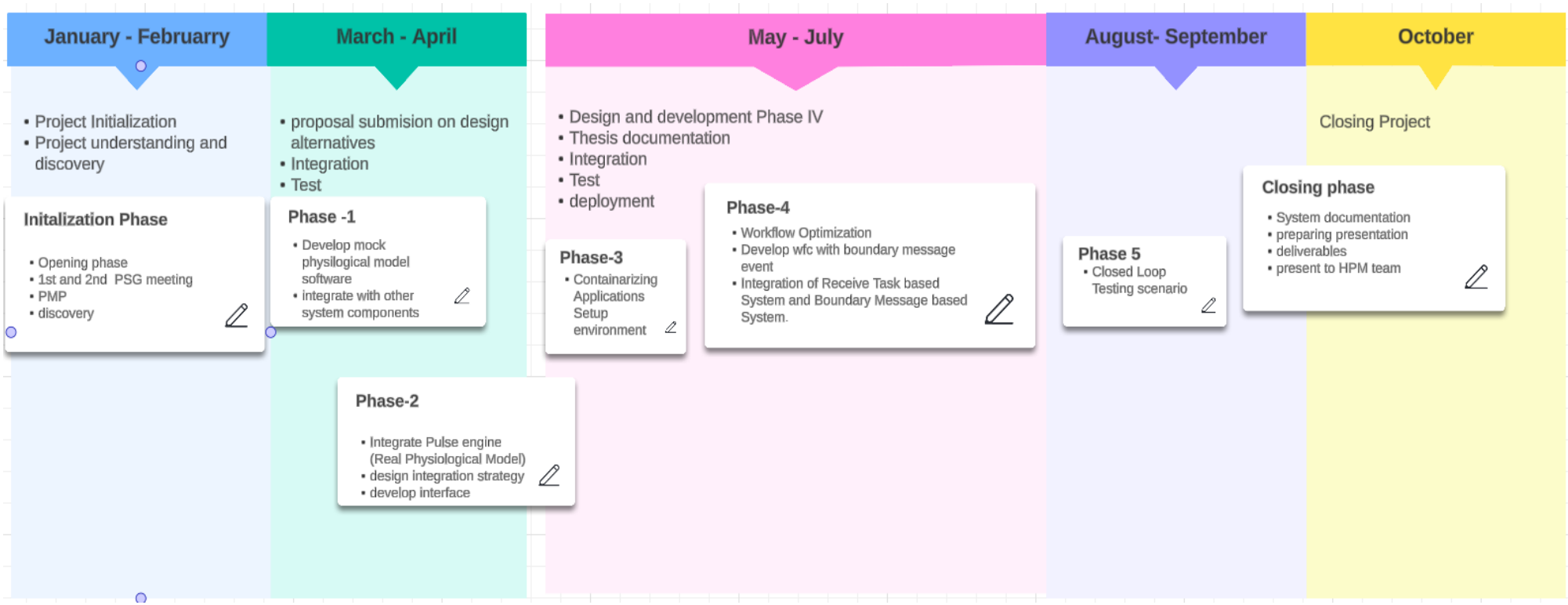

Figure 9.4 Project time

9.4 Risk Management

Effective risk management is crucial in healthcare technology to ensure the successful implementation of systems that directly affect patient care. The HSWC project recognized the necessity of establishing a risk management and mitigation mechanism to address potential challenges that could arise during the integration of the Pulse Physiology Engine.

To assess the feasibility of integrating the Pulse Physiology Engine, we conducted preliminary testing using a mock physiological model. This involved generating random values for vital signs, such as heart rate, blood pressure, and respiratory rate, to create a Virtual Patient. This simulation allowed us to evaluate the system's performance in a controlled environment, providing valuable insights before moving on to the actual integration.

The importance of this testing phase cannot be overstated. By utilizing the mock Virtual Patient, we were able to:

1. **Identify Potential Issues**: The testing highlighted possible challenges that could arise during the integration of the real Physiological Model Software, such as the unavailability of necessary APIs or compatibility difficulties with existing systems.
2. **Mitigate Risks**: By addressing these potential issues early in the process, we could develop strategies to mitigate risks, ensuring a smoother transition to the real integration phase.
3. **Validate Integration Capabilities**: The successful demonstration of the integration with the mock Virtual Patient provided confidence in the HSWC's ability to process and respond to physiological data effectively.

After successfully validating the integration with the mock model, we proceeded to integrate the actual Pulse Physiology Engine. This step was made significantly easier due to the insights gained during the mock testing phase, which allowed us to anticipate and address potential challenges proactively.

Table 9.1 presents the overall risk management plan, including identified risks and their corresponding mitigation strategies.

Table 9.1 Risk management plan

Risk ID	Risk Description	Impact	Likelihood	Mitigation Strategy	Contingency Plan
1	Project Scope (Too Broad)	Delays, loss of focus, inability to deliver a meaningful prototype	Medium	Break down the project into smaller, iterative phases, talk to stakeholder and advisors	Reduce the complexity of the prototype. prioritize tasks; maintain clear communication with stakeholders.
2	Physiological Model Software API- Availability	Unrealistic simulations, inability to evaluate closed-loop system	Low	Thoroughly research existing physiological model options; actively communicate with the lab/team providing the simulator	Develop a simplified "mock" simulator if needed;

3	Integration Complexity	Technical challenges, delays, potential for unstable system	Medium	Phase by phase incremental development	Develop simple API, containerizing some unstable components.
4	HSWC components unexpected error occur	Unexpected errors in HAPI, Camunda and other components	Medium	Thoroughly assess Workflow Capability; initiative-taking communication with the stakeholder	Register and report issues for future enhancements and develop a troubleshooting guide.
5	Stakeholder Alignment	Potential misalignment with stakeholder expectations and project goals	Medium	Maintain regular communication and engagement with stakeholders frequently	Standups and regular update.
6	Technical Debt Accumulation	Future maintenance challenges, increased operational costs	Medium	Implement code reviews and refactoring sessions regularly; enforce coding standards	Plan dedicated sprints for debt reduction and system optimization
7	Document unreadability	Development delay,	High	Use graphs, charts, tables, and good grammar for the client to understand it. Plan to document early	Prepare 2 types of documents, ppt and word. Early presentation about the document
8	Testing Challenges	Inability to conduct unit testing and automation testing due to interconnected components and the codebase lacking unit tests	High	Develop a comprehensive testing strategy that includes creating unit tests for individual components and creating test strategy.	Test important components of the system in a standalone environment. Using E2E testing strategy to ensure all the components behave as expected.

In conclusion, the proactive approach to risk management and the feasibility checks conducted with a mock physiological model were instrumental in the project's success. These steps not only enhanced our understanding of the system's capabilities but also ensured that we were well-prepared to tackle any issues that might arise during the integration with the real physiological model software.

9.5 Project Deliverables

The project deliverables were structured to provide clear outputs at each phase, ensuring that each component not only meets the functional requirements but also aligns with Philips' strategic goals.
Table 9.2 shows the deliverables of the project with its description.

Table 9.2: Deliverables and type of deliverables

Deliverable ID	Type	Description	Remark
D-01	Code	**Code Repositories:** HSWC enhancement code Integrated with Virtual Patient including the BPMN, DMN models and Docker files.	Public/Philips lab GitHub repository

D-02	Code	**PlantUML and diagram:** This includes the code for some diagrams that are developed by PUML tool, and other architectures	Internal/Philips-HPM Microsoft Teams
D-03	Document	**ReadMe document**: Describes the necessary steps to build and deploy and run the HSWC and component level.	Public/Philips-lab GitHub repository
D-04	Document	**Project Management Plan**: This describes how the project will be executed. Containing the project overview, context and planning.	Internal/Philips HPM Microsoft Teams
D-05	Document	**Test Plan and Test Report:** This includes the verification of the product, test cases and expected outcome and the actual report.	Internal/Philips HPM Microsoft Teams
D-06	Document	**Proposal:** Describes about exploring the instrumented clinical workflow modeling methodologies and alternatives.	Internal/Philips HPM Microsoft Teams
D-07	Document	**Final Report and Presentation:** Summarizing the project outcomes, lessons learned, and future recommendations. This includes a PowerPoint presentation for both TU/e and Philips stakeholders.	Internal/Philips and TU/e

Glossary

Workflow	A workflow is a sequence of tasks (clinical tasks) or activities that are organized to achieve a specific goal.
Workflow model	Workflow model is a visual representation of the clinical processes involved in patient care, illustrating the flow of tasks, decision points, and interactions among healthcare providers and systems.
Clinical workflow	Clinical workflow specifically relates to the processes and interactions that occur during patient treatment and care delivery.
Instrumented clinical workflow	An instrumented clinical workflow in this project refers to a clinical workflow that incorporates data integration, allowing for management of tasks. This enables clinical workflows to access critical patient data, such as vital signs, to inform decision-making.
Workflow Engine (BPM Engine)	It is a software application and component of HSWC which is responsible for executing and managing the defined clinical workflows.
Virtual Patient	Virtual Patient refers to a simulated patient model used to test and validate clinical workflows i.e. Physiological Model Software.
Workflow Capability	Core component of HSWC that controls the communication between FHIR store and the BPM engine (workflow engine) to maintain.
HealthSuite Workflow Capability	The software application developed by Philips HPM that contains all the components, Caregiver app, FHIR store, BPM engine or Workflow Engine, WFC, Pulse Engine-FHIR Interface App, and other software application like dashboard.
HAPI FHIR	Healthcare Application Programming Interface - Fast Healthcare Interoperability Resources is an open-source framework for implementing the FHIR standard in healthcare applications. It provides tools for developers to easily create, validate, store, and share healthcare data using FHIR
FHIR	Developed by HL7, is a standard for exchanging healthcare information electronically, using modern web technologies like RESTful APIs.

Bibliography

[1] J. JMJ van der, "Standardized software solution for guidance of clinical workflows," 2021.

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

[3] "agilemania.com," Agilemania Technologies, [Online]. Available: https://agilemania.com/moscow-prioritization-method. [Accessed Oct 2024].

[4] "cflow," Cavintek, Inc, [Online]. Available: https://www.cflowapps.com/workflow-model/. [Accessed 08 2024].

[5] "evidenceCare," evidenceCare, [Online]. Available: https://evidence.care/clinical-workflow-solutions/. [Accessed 09 2024].

[6] K. Zheng, R. M. Ratwani and J. Adler-Milstein, "Studying workflow and workarounds in EHR-supported work to improve health system performance," *Annals of Internal Medicine,* 2020.

[7] "talkinghealthtech," 12 09 2020 . [Online]. Available: https://www.talkinghealthtech.com/glossary/workflows-clinical-workflow. [Accessed 01 09 2024].

[8] "camunda.org," Camunda, [Online]. Available: https://docs.camunda.org/manual/7.21/reference/bpmn20/events/message-events/. [Accessed 09 2024].

[9] O. M. Group, "Business Process Model and Notation (BPMN) Version 2.0.2," 2013.

[10] "Pulse," Kitware, [Online]. Available: https://pulse.kitware.com/. [Accessed 14 May 2024].

[11] "kitware.com," [Online]. Available: https://gitlab.kitware.com/physiology/jupyter/-/blob/master/HowTo_EngineUse.ipynb?ref_type=heads. [Accessed 10 2024].

[12] OMG, "OMG Group," [Online]. Available: https://www.omg.org/about/. [Accessed 14 May 2024].

[13] "HL7," HL7, [Online]. Available: https://info.hl7.org/learn-more. [Accessed 14 May 2024].

[14] "FHIR," [Online]. Available: https://www.hl7.org/fhir/overview.html. [Accessed 14 May 2024].

[15] N. Y. S. S. Z. C. L. L. L. Z. L. J. Li R, "Using Electronic Medical Record Data for Research in a Healthcare Information and Management Systems Society (HIMSS) Analytics Electronic Medical Record Adoption Model (EMRAM) Stage 7 Hospital in Beijing," *JMIR,* 2021.

[16] W. M. P. van der Aalst, "Business Process Management: A Comprehensive Survey," *International Scholarly Research Notices,* 2013.

[17] "https://pretius.com/," pretius, [Online]. Available: https://pretius.com/blog/camunda-bpm/. [Accessed 09 2024].

Appendix A. Workflow Modeling Alternatives

This section presents various instrumented clinical workflow modeling approaches, using a sepsis protocol as a practical use case.

1. Receive Task with Data Object Reference (Existing Model or Base Model)

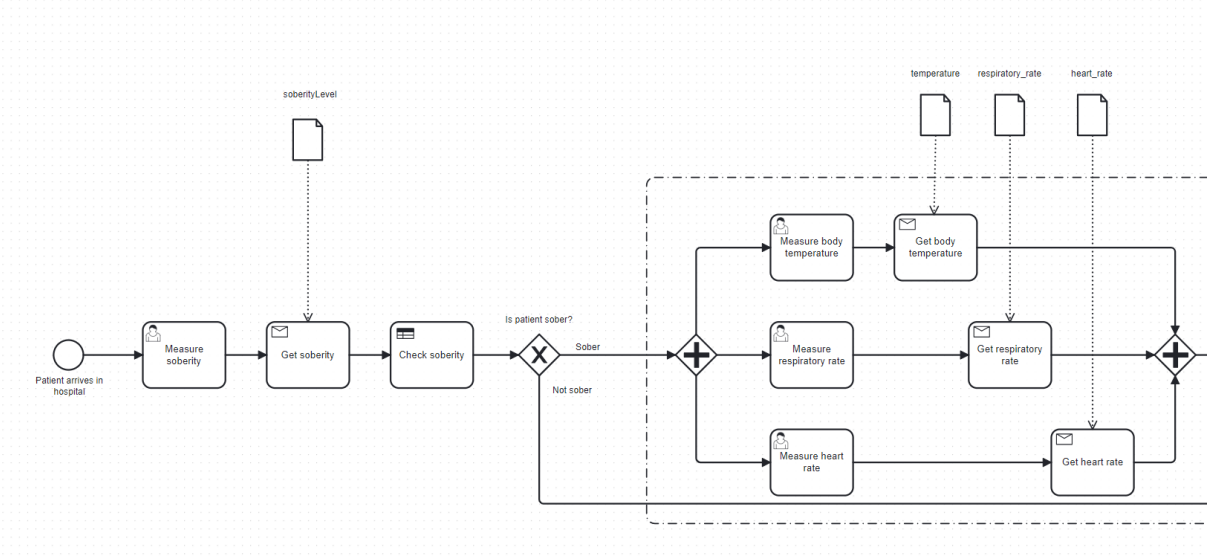

Figure 0.1 Receive Task with Data Object Reference

2. **User Task with Data Object Reference**

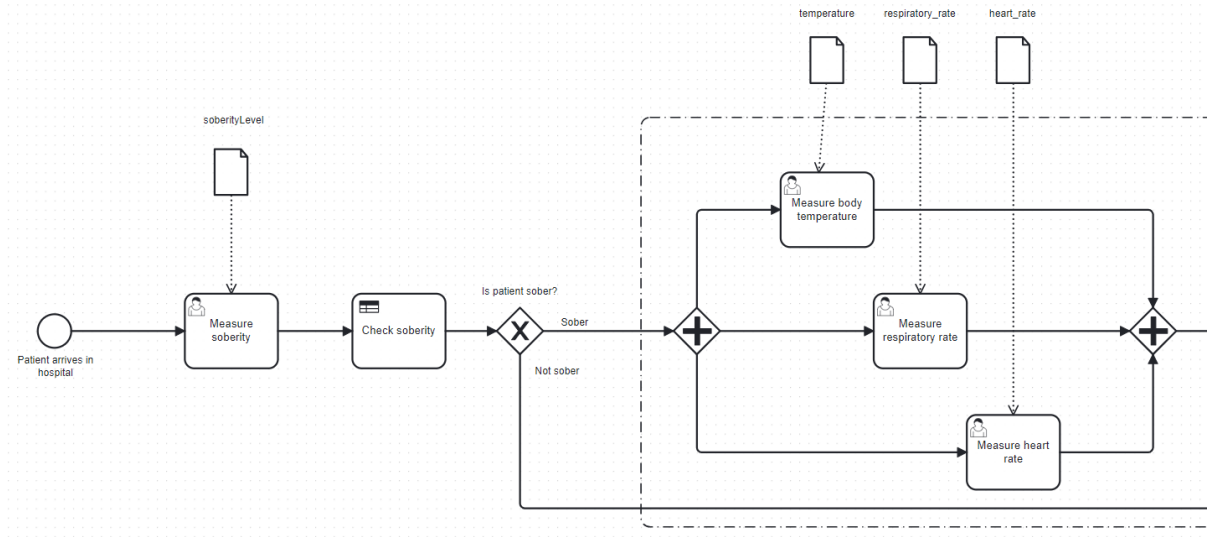

Figure 0.2 User Task with Data Object Reference

3. Using Message Boundary Events (Both interrupting and non-interrupting)

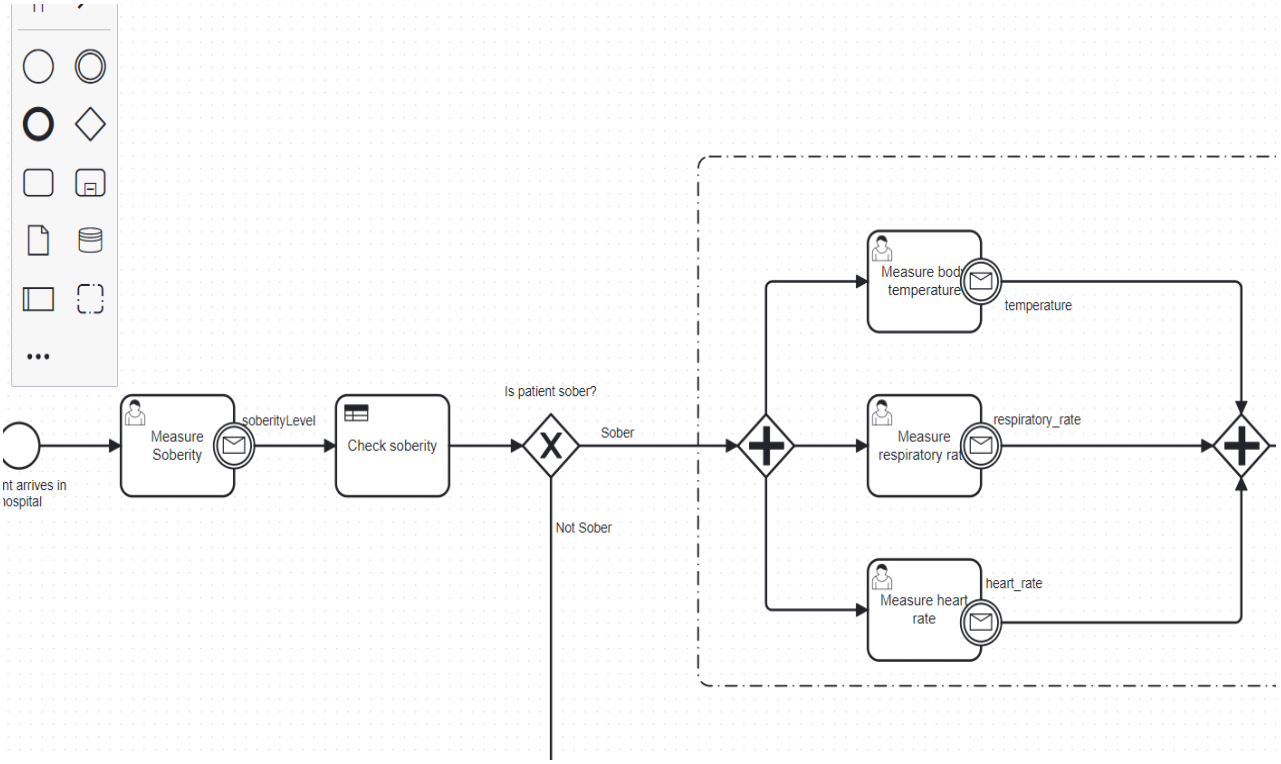

Figure 0.3: User Task with Message Boundary Event

4. Message Intermediate Catch Events

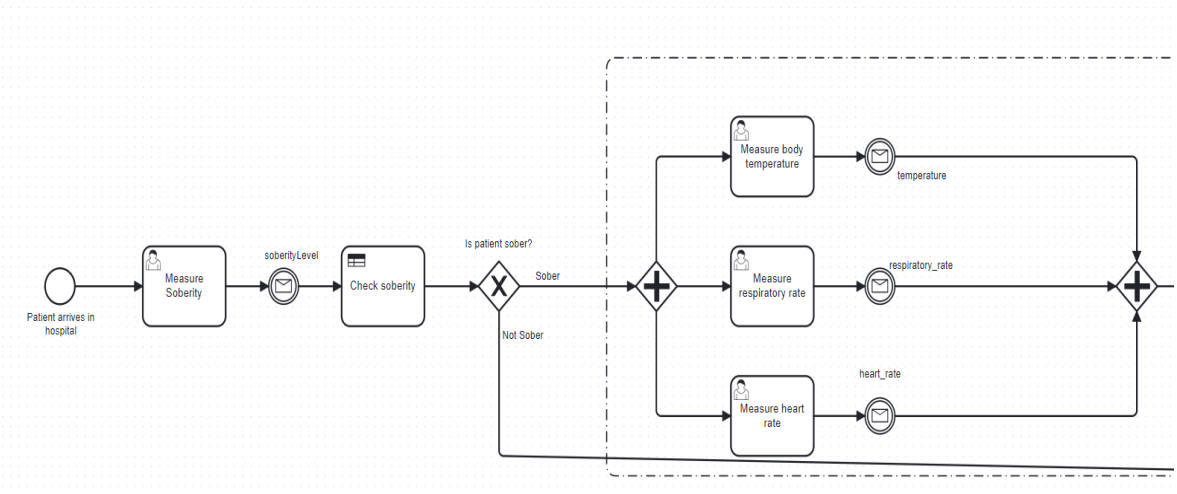

Figure 0.4 Message Intermediate Catch Events

About the Author

Tesfay Gebremeskel Chekole earned his bachelor's degree in Information Technology (Engineering) from Mekelle Institute of Technology, Ethiopia. He then completed a Master's degree in Computer Science, focusing on Intelligent Systems. For his Master's project, he designed a chatbot-based intelligent agent for student counseling and guidance. After graduating, he worked as a lecturer at Mekelle Institute of Technology and as a software engineer at both Metkel Tech and ReNoStar GmbH. Before joining Eindhoven University of Technology, he started another Master's program in Artificial Intelligence and Computer Security at the University of Calabria, Italy. During his EngD at Eindhoven University of Technology, he was involved in projects for Hendrix Genetics, the Applied Data Science Group, and Easy-Go Factory, taking on various technical and non-technical roles. He completed his graduation project for Philips, which forms the basis of this thesis. His current interests include web application development, Machine Learning, particularly in Large Language Models (LLMs).

PO Box 513
5600 MB Eindhoven
The Netherlands
tue.nl

EngD SOFTWARE TECHNOLOGY

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY