UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

VON DER LEITLINIE ZUR ANWENDUNG: EIN FRAMEWORK ZUR SOFTWARE-GESTÜTZTEN UMSETZUNG KLINISCHER PROZESSE

DISSERTATION

DIPL.-INFORM. STEFAN MERSMANN

Aus dem Institut für Telematik der
Universität zu Lübeck
Institutsdirektor: Prof. Dr. Stefan Fischer

VON DER LEITLINIE ZUR ANWENDUNG: EIN FRAMEWORK ZUR SOFTWARE-GESTÜTZTEN UMSETZUNG KLINISCHER PROZESSE

Inauguraldissertation zur Erlangung der Doktorwürde der
Universität zu Lübeck – Sektion Informatik/Technik

vorgelegt von Dipl.-Inform. Stefan Mersmann
aus Warendorf

Lübeck, im Juni 2025

Prüfungskommission:

Vorsitzender:	Prof. Dr.-Ing. habil. Marcin Grzegorzek
1. Berichterstatter:	Prof. Dr.-Ing. Horst Hellbrück
2. Berichterstatter:	Prof. Dr.-Ing. Philipp Rostalski

Tag der mündlichen Prüfung: 18.11.2025

Zum Druck genehmigt. Lübeck, den 23.03.2026

Für Alona. Und mich.

Для Алоны. Мое золотое сердце.

Um die Verständlichkeit und Lesbarkeit dieser Arbeit zu verbessern, verwende ich durchgehend die männliche Form, mit der ich Personen aller Geschlechter gleichwertig ansprechen möchte. Dies geschieht ausschließlich aus Gründen der sprachlichen Vereinfachung und soll keinerlei Ausgrenzung oder Benachteiligung anderer Geschlechtsidentitäten darstellen. Vielmehr soll es verdeutlichen, dass mir ein vielfältiges und inklusives Gesellschaftsbild am Herzen liegt.

VORWORT

1992 bin ich als junger Absolvent der Universität-Gesamthochschule Paderborn und nach einer sehr beeindruckenden und inspirierenden Zeit an der Landesklinik Düsseldorf, an der ich meine Diplomarbeit im Bereich künstlicher Intelligenz in der Hirnforschung verfertigen konnte, zum Medizintechnikhersteller Dräger nach Lübeck gekommen. Hier musste ich zunächst - fernab der künstlichen Intelligenz - Software für Beatmungsgeräte in Assembler programmieren.

1999 jedoch ging es dann los: ein Forschungsprojekt mit der Johannes-Gutenberg Universität in Mainz wollte die gesamte Beatmungstherapie automatisieren. Mit Expertensystemen. Meine Stunde war gekommen.

Von den danach folgenden Stunden und deren spannenden Erträgen der danach folgenden 20+ Jahre berichtet diese Arbeit.

So viele großartige Menschen sind mir auf diesem Weg begegnet. Sie alle aufzuzählen, würde den Rahmen sprengen, den mir mein lieber Doktorvater Horst Hellbrück auferlegt hat. Ihm danke ich von ganzem Herzen für den Mut, einem 60+ Jahre währenden Menschen noch einmal die Faszination der Wissenschaft zu bescheren.

Ach, es muss sein: liebe Smarties, lieber Thomas Katschewitz, lieber Norbert Weiler, lieber Dirk Schädler, lieber Laurent Brochard, lieber Michel Dojat, lieber Volker Belli, lieber Joachim Baumeister, lieber Carsten Wasner, lieber Tim Baier-Löwenstein, lieber Andreas Neumann, lieber Philipp Rostalski, lieber Philippe Jolliet, lieber Philippe Jouvet, lieber Nader Habashi, liebe Penny Andrews, lieber Frank Puppe. Und natürlich all ihr lieben Kolleginnen und Kollegen dieses tollen, norddeutschen Medizintechnikherstellers: Dankeschön!

2025 nun hoffe und wünsche ich, dass vielleicht der eine oder andere Aspekt dieser Arbeit - insbesondere der Schulterschluss zum modernen Prozessmanagement - der heilenden Welt der Medizin von Nutzen sein möge.

Nicht zuletzt gilt mein herzlicher Dank auch all den hier ungenannt bleibenden Personen, die in wertvoller Weise zum Gelingen dieser Arbeit beigetragen haben.

Winseldorf, im Juni 2025

Stefan Mersmann

ZUSAMMENFASSUNG

Der zunehmende gesellschaftliche und politische Druck auf unser Gesundheitssystem, stetig mehr Behandlungsqualität bei gleichzeitig sinkenden Behandlungskosten zu liefern, ist in den letzten Jahrzehnten rasant gestiegen. Dies betrifft auch und im Besonderen die geräte- und pflegeintensiven, medizinischen Disziplinen der Anästhesiologie. Die den täglichen Alltag bestimmenden klinischen Prozesse bieten ein enormes Optimierungspotenzial, wenn sie denn geeignet durch entsprechende Software-Anwendungen unterstützt werden.

Mit dieser Arbeit wird ein generatives, flexibles Software-Framework vorgestellt und bewertet, das die Entwicklung, den Betrieb, die Wartung sowie die Wieder- und Weiterverwendung von Software-Anwendungen für eben solche klinischen Prozesse unterstützt. Im Fokus stehen diagnostische wie auch therapeutische Prozesse als medizinische Leitlinien im Anwendungsbereich der Anästhesiologie. Das Software-Framework selbst ist in Java implementiert und basiert auf einer hybriden, mehrschichtigen Systemarchitektur, die den Einsatz einer virtuellen Maschine unterstützt und damit hohe Flexibilität hinsichtlich Portierbarkeit, Erweiterbarkeit und Wiederverwendung gewährleistet.

Es wird gezeigt, wie klinische Software-Anwendungen zur Entscheidungsunterstützung, halb- und vollautomatisierten Therapie spezifiziert, entworfen, konstruiert, realisiert, verifiziert und validiert und schließlich im hochregulatorischen Umfeld in Verkehr - also in den Markt und damit an den Patienten - gebracht werden können. Entlang eines hybriden Vorgehensmodells werden die zentralen Themen Methodik, Software-Entwicklung und Knowledge Engineering, Technologie, Produktqualifizierung, klinische Evaluierung sowie Zulassung, Inverkehrbringung, Betrieb und Ökonomie eingehend betrachtet und untersucht.

Darüber hinaus ermöglicht die Reflektion auf neuere Ansätze und Technologien aus diesem Bereich aktuelle Ideen und praktische Empfehlungen zu Transfer und Verwertung in gegenwärtigen aber vor allem auch zukünftigen, industriellen Anwendungsszenarien wie auch wissenschaftlichen Forschungsinteressen. Die Untersuchung klassischer Themen, Methoden und Ansätze aus dem modernen Geschäftsprozessmanagement klärt, inwieweit Technologien aus diesem Bereich das Software-Framework zukunftssicherer gestalten können.

ABSTRACT

In recent decades, healthcare systems have faced mounting societal and political pressure to deliver higher quality care while simultaneously reducing treatment costs. This challenge is particularly acute in resource- and care-intensive disciplines such as anesthesiology. Clinical processes that shape daily medical practice offer significant potential for optimization - particularly when effectively supported by tailored software solutions.

This work presents and evaluates a generative, flexible software framework that optimizes the development, deployment, maintenance, and reuse of applications designed to support such clinical processes. The primary focus lies on diagnostic and therapeutic workflows - commonly formalized as clinical guidelines - within anesthesiology. The framework, implemented in Java, is built upon a hybrid, multi-layered system architecture that leverages a virtual machine approach. This design choice enhances the framework's portability, extensibility, and reusability across various clinical contexts.

The work details a full lifecycle of clinical software applications, from specification, design, and implementation to verification, validation, and deployment within a highly regulated environment. These applications support clinical decision-making and facilitate both semi-automated and fully automated therapeutic interventions, ultimately bringing the software into clinical use. Guided by a hybrid development process model, the work systematically explores key dimensions including methodology, software and knowledge engineering, IT infrastructure, product qualification, clinical evaluation, regulatory approval, operational use, and economic considerations.

Additionally, the work reflects on emerging technologies and approaches, offering practical guidance for their integration into current and future industrial applications as well as academic research. Insights from modern business process management further inform how such technologies may enhance the sustainability and forward-compatibility of the proposed software framework.

INHALTSVERZEICHNIS

KAPITEL 1

EINLEITUNG

1.1 MOTIVATION

In den letzten Jahrzehnten ist der gesellschaftliche und politische Druck auf das Gesundheitssystem signifikant gestiegen. Die Forderungen nach einer kontinuierlichen Steigerung der Behandlungsqualität bei gleichzeitiger Reduzierung der Behandlungskosten sind intensiviert worden. Dieser Druck führt zu einer komplexen Herausforderung, bei der innovative Lösungen und effektive Strategien erforderlich sind, um die nachhaltige Erfüllung dieser Anforderungen sicherzustellen. Die Balance zwischen wirtschaftlicher Effizienz und medizinischer Exzellenz wird zunehmend kritisch betrachtet, wobei der Fokus auf der Optimierung von Prozessen, der Implementierung neuer Technologien und der Anpassung von politischen Rahmenbedingungen liegt, wie eine Studie der Organisation für wirtschaftliche Zusammenarbeit und Entwicklung (OECD) 2023 herausgefunden hat [87].

Es ist von einer Kostenexplosion im Gesundheitswesen die Rede. Das statistische Bundesamt [122] teilte 2024 dazu mit:

„Die Gesundheitsausgaben in Deutschland sind im Jahr 2022 gegenüber dem Vorjahr um 4,8 % oder 22,6 Milliarden Euro auf 497,7 Milliarden Euro gestiegen. Das waren 5 939 Euro je Einwohnerin und Einwohner. Wie das Statistische Bundesamt (Destatis) weiter mitteilt, lag der Anteil der Gesundheitsausgaben am Bruttoinlandsprodukt (BIP) 2022 bei 12,8 % und damit 0,3 Prozentpunkte niedriger als 2021."

Soll heißen: Die Kosten für unsere Gesundheit steigen kontinuierlich, gleichzeitig jedoch sinkt der Anteil am Bruttosozialprodukt. Neben dem Kostenfaktor verdient aber auch ein weiterer, sozio-psychologischer Aspekt nicht mindere Beachtung: die Situation der im Gesundheitswesen tätigen Menschen, die gekennzeichnet ist durch Stress, Überlastung, Personalmangel sowie weiteren Konfliktpotenzialen und sozialen Begleiterscheinungen. Dies führt laut Marckmann und Schildmann (2022) immer häufiger zu Einbußen in der Behandlungsqualität und daraus resultierenden Behandlungsfehlern [72].

Insgesamt stellt solch eine Entwicklung unser Gesundheitswesen in jeder Hinsicht vor enorme Herausforderungen. Und auch die Industrie - speziell die Medizintechnik - hat erkannt, dass diese defizitäre Situation viele Chancen auf dem Weltmarkt bietet. Es gilt, jene Ambivalenz zwischen Behandlungsqualität, Behandlungskosten und der Mitarbeiterzufriedenheit stets im Fokus zu behalten.

Zudem verschärft ein weiteres Dilemma diese Problematik erheblich: die zunehmenden gesetzlichen und regulatorischen Anforderungen an die Entwicklung von Medizinprodukten. Derlei Rahmenbedingungen stehen der dringend benötigten Innovation bisweilen im Wege. So haben sich die bürokratischen Auflagen und damit der Dokumentationsaufwand für Entwicklungsprojekte mit der Ablösung des Medizinproduktegesetzes (MPG) durch die Medizingeräterichtlinie (MDR) im Jahre 2017 massiv erhöht. Dies betrifft auch und besonders die geräte- und pflegeintensiven Fachdisziplinen der Anästhesiologie.

Das Institute for Healthcare Improvement (IHI) hat 2008 eine großangelegte Kampagne gestartet, mit der drei zentrale, strategische Säulen und deren Abhängigkeiten zueinander adressiert werden: Triple-Aim [62], siehe Abbildung 1.1. Grundgedanke dabei ist, dass wenn Bemühungen im medizinischen und/oder medizinnahen Umfeld die Schwerpunkte *Gesundheit der Bevölkerung, Behandlungserfahrung* und *Pro-Kopf-Kosten* zum erklärten Ziel haben, dann lässt sich obiges Dilemma bezwingen und Gesundheitssysteme verbessern.

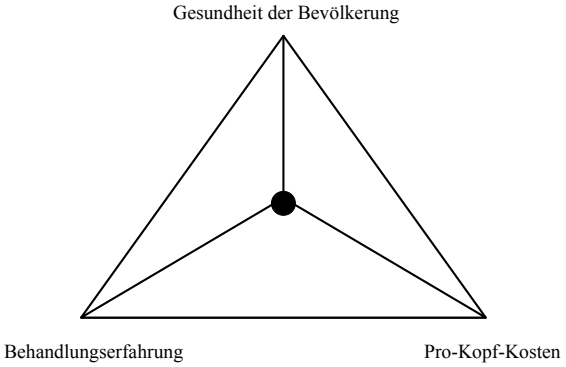

Abbildung 1.1 – *Die drei großen Ziele der IHI*

Erfreulicherweise hat sich das Triple-Aim inzwischen um ein weiteres Ziel zum Quadruple-Aim erweitert, das nun auch die *Zufriedenheit des Klinikpersonals* mit einbezieht. Viele Medizintechnikhersteller nutzen bereits das Quadruple-Aim zur Gestaltung von Geschäftsstrategie und Produktportfolio und auch die vorliegende Arbeit adressiert diese Ziele ausdrücklich.

1.2 PROBLEMSTELLUNG

Diagnostik und Behandlungsqualität zu erhöhen und gleichzeitig Behandlungskosten zu reduzieren stellt generell eine große Herausforderung dar. Aus Sicht der Medizintechnikbranche, die für diese Herausforderung geeignete Produkte und Dienstleistungen entwickeln und vermarkten will, ist ein vielversprechender Ansatz, Abläufe im klinischen Alltag zu untersuchen. Sind diese Abläufe dann identifiziert und verstanden, kann ermittelt werden, wie und in welchem Umfang diese durch Software-Anwendungen unterstützt werden können.

Oftmals sind Software-Anwendungen für genau einen Einsatzbereich und für genau ein Zielsystem (i. e. Medizinprodukt) ausgelegt. Auch fehlt die Synthese von Verfahren, Methoden, Technologien und Werkzeugen, vor allem zur autonomen Steuerung von Medizingeräten. Erst eine „Fernsteuerung" der eingebundenen Medizingeräte ermöglicht eine vollumfängliche Unterstützung klinischer Abläufe.

Im Sinne der Flexibilität, Wieder- und Weiterverwendung und damit der Zukunftssicherung, sollten möglichst viele klinische Abläufe und gleichzeitig möglichst viele Zielsysteme versorgt werden können. Diese Forderung wird umso dringlicher, je genauer wir uns Fülle und Komplexität klinischer Abläufe und daran beteiligter Medizingeräte vor Augen führen. In dieser Arbeit geben wir praktische Beispiele, die dies verdeutlichen.

Eine weitere Herausforderung ist die Häufigkeit und der Umfang, in dem sich medizinisches Wissen ändern kann. Wenn wir in Software-Anwendungen eingebettetes medizinisches Wissen einfach und schnell - idealerweise „auf Knopfdruck" - aktualisieren können, dann wird dieser Änderungsprozess nennenswert optimiert. Zu den regelmäßigen Änderungen zählt auch die erneute Produktzulassung, denn Software-Anwendungen im klinischen Einsatz unterliegen gesetzlichen, regulatorischen Anforderungen. Führen Änderungen zu einer neuen Version, so müssen auch die jeweils gültigen regulatorischen Anforderungen erneut geprüft und nachgewiesen werden.

Im medizinischen Umfeld haben Bemühungen zur Standardisierung und Harmonisierung von klinischen Abläufen zum Konzept der medizinischen Leitlinien, engl. Clinical Guideline (CG), geführt. Die Arbeitsgemeinschaft der Wissenschaftlichen Medizinischen Fachgesellschaften (AWMF) definiert medizinische Leitlinien wie folgt:

Definition MEDIZINISCHE LEITLINIE

> Leitlinien sind systematisch entwickelte Aussagen, die den gegenwärtigen Erkenntnisstand wiedergeben, um die Entscheidungsfindung von Ärzten sowie Angehörigen von weiteren Gesundheitsberufen und Patienten/Bürgern für eine angemessene Versorgung bei spezifischen Gesundheitsproblemen zu unterstützen [55].

Mittlerweile sind manuell anzuwendende Leitlinien für nahezu alle medizinischen Anwendungsgebiete verfügbar. So bietet beispielsweise die World Health Organisation (WHO) in ihrem Archiv eine Vielzahl medizinischer Leitlinien an [124] und Not-for-Profit-Organisationen wie das Guideline International Network (GIN) [118] mit Zugang aus über 100 Ländern oder deren deutsches Gründungsmitglied, die Ärztliche Zentralstelle für Qualitätssicherung in der Medizin (ÄZQ) [113], beschäftigen sich intensiv mit Aufbau und Pflege von Datenbanken für medizinische Leitlinien.

Medizinische Leitlinien bieten nachweislich zahlreiche Vorteile, die u. a. von Ely et al. [67], Marelich et al. [74] und Burns et al. [97] in klinischen Studien und/oder Untersuchungen herausgearbeitet wurden:

- Steigerung der Behandlungsqualität
- Reduktion von Behandlungskosten und -dauer
- Standardisierung von Abläufen
- Entlastung des Personals
- Identifizierbarkeit von Verbesserungspotenzialen
- Fehlervermeidung bzw. –reduzierung
- Reduzierung der Sterberate
- Reduzierung von Komplikationen, die durch Medizingeräte hervorgerufen werden

Für Details verweisen wir auf die Ausführungen von Ely, Marelich und Burns. Die in dieser Arbeit vorgestellte Lösung beschäftigt sich mit der computergestützten Umsetzung medizinischer Leitlinien, womit sie sämtliche oben genannten Vorteile bieten kann.

In der Praxis finden wir medizinische Leitlinien überwiegend in Papierform. Sei es als Notizzettel in der Kitteltasche, Checkliste im Aufenthaltsraum, laminiertes Flussdiagramm am Patientenbett oder in klinikintern veröffentlichter Buchform. Auch ist der Grad der Harmonisierung dieser Papierderivate sehr unterschiedlich. Das reicht von individueller Erfahrung über Konsens im Team bis hin zu klinikweiten Vorgaben. Dementsprechend ist die Anwendung medizinischer Leitlinien am Patientenbett äußerst aufwändig, zeitintensiv, mühsam und fehlerbehaftet.

Außerdem ist Standardisierung, effiziente Repetition, einheitliche Dokumentation und besonders eine verlässliche, quantitative Leistungsmessung klinischer Abläufe nahezu unmöglich. Gerade Letztere wird jedoch dringend für ein Benchmarking im Sinne eines kontinuierlichen Verbesserungsprozesses im Qualitätsmanagement eines Krankenhauses immer eindringlicher gefordert. Das haben zahlreiche Gespräche mit Ärzten und Pflegepersonal während Konferenzen, Tagungen und Klinikbesuchen gezeigt.

Mit der vorliegenden Arbeit wollen wir dieser Problematik begegnen, indem wir medizinische Leitlinien computerisieren und Ärzten und Pflegepersonal in Form von Software-Anwendungen verfügbar machen. Dieser Ansatz ist nicht gänzlich neu, wie vergleichbare Forschungsarbeiten zeigen, und wird mit dieser Arbeit in Richtung Automatisierung fortgeführt.

1.3 VERWANDTE ARBEITEN

Um die Jahrtausendwende erlebte das Thema „Computerisieren von medizinischen Leitlinien" ein verstärktes Interesse in der Wissenschaftswelt. Es ging und geht um die Standardisierung klinischer Abläufe, Vermeidung von Behandlungsfehlern und Entlastung des Klinikpersonals.

Zwei Themenschwerpunkte kristallisierten sich dabei heraus:

- die Repräsentation medizinischer Leitlinien im Computer und
- die Entwicklung sogenannter Autorensysteme

Während eines Workshops im Columbia's Arden Homestead Conference Center im Juni 1989 entstand die sogenannte Arden-Syntax, mit der medizinisches Wissen, insbesondere medizinische Leitlinien, in einfacher, menschenlesbarer Form abgebildet werden kann [100]. So entwickelte sich das Repräsentationsformat der Medical Logic Modules (MLM), mit denen sowohl ein Austausch medizinischen Wissens, aber auch die Weiterverarbeitung desselben mit dem Computer angestrebt wurden.

In der Arden-Syntax verfasste MLM liegen im ASCII-Format vor und können daher mit jeglicher Art von Texteditor erstellt werden. Sie gleichen eher einer Programmiersprache und sind daher nur bedingt für den Austausch mit Medizinern geeignet.

Ein Beispiel für ein kurzes MLM ist in Algorithmus 1.1 zu sehen. Das MLM-Fragment überwacht den Wert der Glukose (Blutzucker), indem es ihn mit einem Schwellwert vergleicht und bei Unterschreitung einen Alarm zum Patientenmonitor sendet. MLM ist eine einfache Programmiersprache, in der Logik- und Aktionsblöcke sequentiell aneinander gereiht werden.

```
1 LOGIC:
2 IF glucose IS LESS THAN 50 THEN
3 CONCLUDE true;
4 ENDIF;
5 ACTION:
6 WRITE alert AT patientmonitor;
```

Algorithmus 1.1 – *MLM zur Glukoseüberwachung*

Das einige Jahre später entstandene Guideline Interchange Format (GLIF) [81] ist ein weiteres Repräsentationsformat für medizinische Leitlinien, das deutlich umfangreicher ist als die Arden-Syntax. GLIF definiert eine Leitliniensystematik bestehend aus Modell und Syntax, mit der sich modular aufgebaute Leitlinien-Repräsentationen erstellen lassen.

Als Beispiel dazu ist in Algorithmus 1.2 ein wiederverwendbares GLIF-Modul zur Verabreichung eines generischen Impfstoffes X aufgeführt.

```
1  {
2      name = "Guideline for Vaccine X"
3      authors = SEQUENCE 1 { "Mary Doe, MD";};
4      eligibility_criteria = NULL;
5      intention = "Decide whether to recommend Generic vaccine
           and at what dosage";
6      steps =
7          SEQUENCE8
8          {   {Branch_Step 1};            {Action_Step 1};
                   {Action_Step2};
9              {Synchronisation_Step 1};   {Conditional_Step 1};
                       {Conditional_Step 2};
10             {Action_Step 3};            {Action_Step 4};
11         };
12     first_step = {Branch_Step 1};
13     didactics =
14         SEQUENCE 1
15         {
16             Supplemental_Material 1
17             {   label = "critique";
18                 MIME_TYPE = "text/plain";
19                 material = "Published guideline does not
                       contain explicit eligibility_criteria";
20             };
21         };
22 }
```

ALGORITHMUS 1.2 – *GLIF-Modul zur Impfmittelgabe*

GLIF-Exponate sind immer noch sehr technisch und programmatisch, bieten jedoch weiterreichendere Konstrukte als MLM. So können Module, Funktionen und Verschachtelungen gebildet, Texte verarbeitet und Multimediadaten eingebunden werden. Auch stehen mehr und komplexere Logikoperatoren zur Verfügung als in MLM.

Verschiedene Ansätze vereinfachen die Erstellung von GLIF-Modulen beispielsweise durch graphische Flussdiagramme, die in das GLIF übersetzt werden. Mit GLIF3 wurde darüber hinaus eine Weiterentwicklung vorgestellt, die die Unified Modelling Language (UML) für die graphische Darstellung eines GLIF-Modells nutzt [104].

Mit der Forderung, computerisierte Leitlinien auch den Medizinern zugänglicher zu machen, um damit eine effizientere Kooperation zu ermöglichen, entstanden diverse, sogenannte Autorensysteme. Diese sind vergleichbar mit Entwicklungsplattformen, die die Erstellung und Pflege computerbasierter medizinischer Leitlinien vielfältig unterstützen.

Im Forschungsprojekt Asgaard wurde ein sehr mächtiges Autorensystem vorgestellt [101], das sich kontinuierlich weiterentwickelt hat. Die Autoren sprechen bereits von einem Framework für eine Leitlinien-Bibliothek. Zusätzlich zu einem syntaktischen Repräsentationsformat namens Asbru beschäftigt sich Asgaard vor allem mit der Modellierung von zeitorientierten Entitäten, genauer: Zeitintervallen. Diese sind für die Darstellung klinischer Abläufe unabdingbar und erlauben die Modellierung von Ereignissen wie beispielsweise „die Insulingabe startete

24h vor Blutentnahme". Es existieren zahlreiche graphische Front-Ends für die komfortable Bearbeitung von in Asbru verfassten medizinischen Leitlinien.

Einen guten Überblick über weitere Autorensysteme gibt das Review von Declercq et al. [91].

Allen hier erwähnten, akademischen Ansätzen gemein ist, dass sie über die reine Modellierung hinaus auch die Ausführung medizinischer Leitlinien gestatten. Diese beschränkt sich allerdings ausschließlich auf den Ablauf in einem solitären Computer, der nicht mit etwaiger Medizintechnik interagiert, sondern isoliert ist und manuell bedient wird. Es erfolgt also weder eine Datenaufnahme noch eine autonome Steuerung von beteiligten Medizinprodukten. Doch gerade dieser Aspekt ist entscheidend für eine vollumfängliche Unterstützung des Klinikpersonals durch entsprechende Software-Anwendungen. Er bildet einen der zentralen Schwerpunkte dieser Arbeit.

1.4 LÖSUNGSANSATZ

1999 erging bei der Firma Dräger in Lübeck der Entwicklungsauftrag, ein akademisches, prototypisches System in ein bestehendes Beatmungsgerät zu portieren und zu kommerzialisieren. Es handelte sich um das im Rahmen einer Dissertation an der Universität Paris entwickelte NeoGanesh [83], mit dem die Entwöhnung eines Patienten von der maschinellen Beatmung partiell automatisiert wurde.

Während der Analyse des NeoGanesh-Systems entstand die Idee, nicht nur eine 1:1 Portierung vorzunehmen, sondern vielmehr ein Software-Framework (SWFW) zu schaffen, mit dem auch weitere Software-Anwendungen dieser Art effizient und flexibel produziert werden können (m:n). 1:1 meint in diesem Zusammenhang: ein klinischer Ablauf (z. B. Entwöhnung von der Beatmung) für ein Zielsystem (z. B. Evita 4), wohingegen n:m bedeutet: viele klinische Abläufe (m) für viele Zielsysteme (n).

Dies führte zum Konzept eines generativen, anpassbaren Software-Frameworks für die Modellierung und Ausführung medizinischer Leitlinien in vielen Medizinprodukten. Unter einem Software-Framework verstehen wir in diesem Zusammenhang:

Definition SOFTWARE-FRAMEWORK

Ein Programmiergerüst, das aus einer Sammlung von wiederverwendbaren Entwicklungsprozessen, Paradigmen, Design-Prinzipien, Werkzeugen, Modellen, Konventionen, Software-Komponenten und Dokumentation besteht, die die Entwicklung und Wartung neuer Software-Anwendungen flexibler und effizienter machen.

Das übergeordnete Leitprinzip des Software-Frameworks legen wir fest als:

Vom Programmieren zum Spezifizieren

Gemeint ist damit die Bestrebung, einen möglichst hohen Anteil der Entwicklungsarbeit mit dem Spezifizieren der Software-Anwendung - also einer medizinischen Leitlinie - abzudecken, um so möglichst viel der generierenden Funktionalität des Software-Frameworks zu überlassen. Damit folgt unser Leitprinzip dem Paradigma der deklarativen Programmierung, das den Fokus auf das *Was* - das gewünschte Ergebnis - statt auf das *Wie* - den konkreten Ablauf - legt.

Darüber hinaus ist das fundamentale und durchgängige Design-Prinzip die strikte Trennung von invarianten und variablen Anteilen, sodass lediglich die variablen Anteile dem Änderungszyklus unterliegen und die invarianten im Sinne eines Frameworks wieder- und weiterverwendet werden können. Das mit den jeweiligen Software-Anwendungen umgesetzte medizinische Wissen stellt typischerweise einen solchen variablen Anteil dar, wohingegen beispielsweise Problemlösungsstrategie, Interpretation und Schlussfolgerungen aus dem medizinischen Wissen sowie spezifische technische Komponenten unverändert bleiben.

Ebenso fundamental und durchgängig sind drei Paradigmen, die gleichsam den Lösungsraum des Software-Frameworks aufspannen:

1. Zwei Freiheitsgrade
2. Zweiphasiger Nutzungszyklus
3. Hybrides Vorgehensmodell

Diese drei Paradigmen definieren die Herausstellungsmerkmale des Software-Frameworks und bestimmen, wie die zentralen Anforderungen an das System nachhaltig und zukunftssicher umgesetzt werden. Ihnen ist jeweils ein eigener Abschnitt im Kapitel 4 gewidmet. Realisiert werden sie mit folgenden Ansätzen und Technologien.

Eine mehrschichtige Systemarchitektur, die eine virtuelle Maschine verwendet und leicht um beliebige Komponenten erweiterbar ist, setzt das Design-Prinzip sowie das Paradigma der zwei Freiheitsgrade um, welches zum Ziel hat, sowohl viele medizinische Leitlinien (Freiheitsgrad 1) als auch viele Medizingeräte (Freiheitsgrad 2) mit entsprechenden Software-Anwendungen versorgen zu können. Diese Systemarchitektur wurde erstmalig 2001 von Mersmann und Dojat auf einem Symposium in Aalborg, Dänemark präsentiert [21].

Das Paradigma des zweiphasigen Nutzungszyklus bestehend aus Modellierung und Ausführung klinischer Prozesse ist mit einem adaptierten Knowledge Wiki als kollaborative, integrierte Entwicklungsplattform verwirklicht, das 2012 von Hatko et al. vorgestellt wurde [18], [20]. Zu dieser Entwicklungsplattform gehört auch eine funktionale Erweiterung nebst syntaktischer Notation, mit der jedwede Prozesse grafisch darstellbar sind, speziell klinische Prozesse. Diese grafische Notation ist beispielsweise in Hatko et al. beschrieben [105].

Die hochinteraktive Zusammenarbeit zwischen medizinischen Experten und Software-Entwicklern wird zusätzlich zur Entwicklungsplattform durch die De-

finition eines hybriden Vorgehensmodells, dem dritten Paradigma, fortgeführt, siehe auch Mersmann et al. [5], [6].

Im Abschnitt 1.3 wurden einige Ansätze zur Computerisierung medizinischer Leitlinien vorgestellt. Sie alle vereint die Eigenschaft, dass sie die an den klinischen Abläufen beteiligten Medizingeräte ausschließlich als Daten- und Informationsquelle (Read-only) nutzen und damit ein sogenanntes Open-Loop-System bilden.

Im Gegensatz zu einem Open-Loop-System schließt ein Closed-Loop-System den funktionalen Wirkungskreis, indem auch Änderungen an den beteiligten Medizingeräten möglich sind. Es handelt sich also um eine Art autonome Steuerung zum Zwecke der automatisierten Therapieführung, vergleichbar in etwa mit einem Autopiloten (Read-Write).

Ohne die explizite Nutzung der autonomen Steuerung (i. e. Vollautomatik) entstehen Open-Loop-Anwendungen, deren Ziel die Entscheidungsunterstützung ist, sogenannte Clinical Decision Support Systems.

Das Software-Framework unterstützt sowohl den Open-Loop- als auch den Closed-Loop-Betrieb. Obendrein ermöglicht es den halbautomatischen Betrieb, bei dem die behandelnde Person von der Software-Anwendung ermittelte Einstellungsveränderungen des Medizingerätes manuell bestätigen muss, bevor diese wirksam werden.

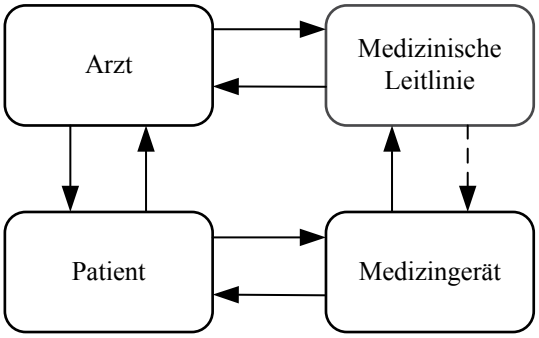

Abbildung 1.2 – *Daten- und Kontrollflüsse der beteiligten Akteure*

In Abbildung 1.2 sind die Akteure Arzt, Patient, Medizingerät und medizinische Leitlinie sowie deren Daten- und Kontrollflüsse untereinander illustriert. Der Closed-Loop-Betrieb (Halb- und Vollautomatik) ist gekennzeichnet durch den gestrichelten Pfeil, über den sich die autonome Steuerung des Medizingerätes vollzieht. Mit Wegfall der gestrichelten Verbindung entsteht der Open-Loop-Betrieb zur Entscheidungsunterstützung, die ohne autonome Steuerung auskommt. Mit dem Software-Framework erstellte Anwendungen können somit zwischen drei Betriebsarten wählen: Entscheidungsunterstützung, Halbautomatik oder Vollautomatik.

Das Ende der Implementierung unserer Lösung kennzeichnet die wichtigen Aktivitäten zur Evaluierung von Software-Framework und damit entwickelte Software-Anwendungen. Die sorgfältige Auswahl geeigneter Evaluierungsverfahren sowie

die Entwicklung eigener Test-Software, die auf die Spezifika der Patient-Medizinprodukt-Verbindung eingehen, stehen hier im Vordergrund [10]. Zur Evaluierung zählt aber auch die technische und wissenschaftliche Begleitung und Unterstützung klinischer Studien, mit denen Wirksamkeit und Sicherheit der Entwicklungsergebnisse geprüft werden. Im Rahmen der Entwicklung, des Betriebs und der Weiterentwicklung des Software-Frameworks haben wir diverse klinische Studien begleitet, deren Ergebnisse u. a. in [3], [53], [90], [15] und [73] publiziert wurden und die wir auch im Verlauf der Arbeit noch detailliert vorstellen werden.

Die erste Implementierung, Bereitstellung und Nutzung des Software-Frameworks stellt quasi die erste Generation dar. Aus den Ergebnissen der Untersuchung potenzieller Methodiken und Technologien aus dem Bereich des Prozessmanagement kann eine zweite Generation erwachsen.

1.5 WISSENSCHAFTLICHE BEITRÄGE

Die wichtigsten, eigenen wissenschaftlichen Beiträge des Autors sind themenorientiert in einer Reihenfolge entlang des Lösungsweges angeführt und folgen damit Gliederung und Struktur des Textes.

Mitsamt den korrespondieren Publikationen sind dies:

Die Konzeptionierung und Entwicklung einer Methodik und eines Software-Framework zur flexiblen Konstruktion, Ausführung und Pflege von Software-Anwendungen [22], [4], [7].

Der Entwurf, die Implementierung und die weiterführende Nutzung einer flexiblen, generativen Systemarchitektur [21], [16].

Die Erstellung und Etablierung eines hybriden Vorgehensmodells zur Synthese von Software-Engineering und Wissensmodellierung [5], [6].

Die Verwendung von Knowledge Wikis als Entwicklungsplattform und universelle Datenquelle für wissensbasierte Systeme [18], [20].

Die grafische Notation DiaFlux zur Modellierung und Untersuchung von komplexen, klinisch-therapeutischen Prozessen innerhalb eines Knowledge Wiki [105].

Evaluierung, Verifikation und Validierung wissensbasierter System anhand reaktiver in silico Testumgebungen [10], [2].

Die klinische Evaluierung wissensbasierter Software-Anwendungen, welche mit dem Software-Framework entwickelt wurden [3], [8], [17], [15].

Mit dem Software-Framework entwickelte kommerzielle und akademische Beispielanwendungen aus dem klinischen Alltag sowie wissenschaftliche Untersuchungen dazu [5], [14], [13], [1], [12], [11].

1.6 AUFBAU DER ARBEIT

Nach dieser Einleitung wird im Kapitel 2 der Anwendungsbereich Anästhesiologie für mit dem Software-Framework erstellte Software-Anwendungen näher bestimmt und abgegrenzt. Dazu gehört auch die weitere Verfeinerung medizinischer Leitlinien und deren unterschiedliche Ausprägungen. Außerdem führen wir den Begriff der klinisch-therapeutischen Prozesse ein, mit dem eine Konkretisierung auf eine methodische und technische Umsetzung möglich wird. Die Vorstellung eines generischen Behandlungsprozesses, der jedem klinisch-therapeutischen Prozess innewohnt, sowie Beispielprozesse allgemeiner Art und aus dem klinischen Umfeld schließen das Kapitel.

Fundament und Ausgangspunkt für Methodik und Technologie des Software-Frameworks sind wissensbasierte Systeme. Sie werden mitsamt ihrer umschließenden Thematik der Expertensysteme als Teildisziplin der künstlichen Intelligenz im Kapitel 3 vorgestellt. Dazu gehört auch die Auflistung und Erklärung von Problemlösungsstrategien, wie sie in Expertensystemen als zentrale Logikinstanz zum Einsatz kommen. Schließlich beschreiben wir an dieser Stelle sogenannte Knowledge Wikis als Spezialisierung wissensbasierter Systeme, die dem kollaborativen Ansatz folgen.

In Kapitel 4 stellen wir ausführlich die Methodik des Software-Frameworks und deren Besonder- und Neuheiten vor. Neben dem Anforderungsmanagement gemäß eines wohlstrukturierten Software-Entwicklungsprozesses, spielt die Systemarchitektur eine übergeordnete Rolle für die Zusicherung geforderter Qualitätsattribute wie Portierbarkeit, Robustheit, Gebrauchstauglichkeit etc. Schließlich werden die drei zentralen Paradigmen im Einzelnen vorgestellt und detailliert.

Die Methodik soll unabhängig von der jeweiligen Technologie sein. Letztere wird in Kapitel 5 beschrieben. Nach der Festlegung der verwendeten Entwicklungsumgebung erläutern wir die konkrete Implementierung mitsamt ihren Besonderheiten. Insbesondere der Transfer der modellierten Prozesse in eine Ausführungsumgebung, die Dynamik des Gesamtsystems sowie die Anbindung an Zielsysteme stehen hierbei im Vordergrund.

Das Software-Framework selbst sowie die damit entwickelten wissensbasierten Software-Anwendungen sind in der Regel als Medizinprodukte klassifiziert und unterliegen daher gesetzlichen und regulatorischen Anforderungen. Deren Einhaltung sichert und überprüft das Qualitätsmanagement, sodass letztendlich eine Produktzulassung und die Markteinführung möglich wird. Kapitel 6 erklärt die Gesetzes- und Normenlage sowie das notwendige Prozedere, von der Verifikation und Validierung über die klinische Evaluierung bis hin zur Produktzulassung.

Beispiele für wissensbasierte Software-Anwendungen für die Anästhesiologie, die mit dem Software-Framework entwickelt wurden und es als Medizinprodukte in den Markt geschafft haben, finden sich im Kapitel 7. Aber auch jene, die akademischer Natur geblieben sind und/oder zu Forschungszwecken für dedizierte Fragestellungen entstanden, sind hier aufgeführt.

Während wir bis hierhin die Implementierung des Software-Frameworks als wissensbasiertes System - als erste Generation - beschrieben haben, unternehmen

wir nun den Schritt in Gegenwart und Zukunft - der zweiten Generation. So untersuchen wir in Kapitel 8, inwieweit Erkenntnisse, Methoden und Praktiken aus dem Bereich des konventionellen Geschäftsprozessmanagement in der Lage sind, ein Software-Framework, wie es zuvor vorgestellt wurde, nach zu implementieren bzw. partiell zu substituieren. Dabei finden die beiden anfangs eingeführten Prozessbeispiele Anwendung.

Kapitel 9 schließlich fasst die Ergebnisse und Erkenntnisse dieser Arbeit zusammen, liefert eine Rückschau im Sinne eines „Lessons learned" und zeigt Potenziale für Optimierung und Weiterführung dieses spannenden Themas.

Wo immer relevant beginnt jedes Kapitel mit einer Einordnung in Forschung und Technik und stellt die wissenschaftlichen Neuheiten dieser Arbeit abgrenzend zum gegenwärtigen Stand von Fachwelt und Industrie dar. Darüber hinaus endet jedes Kapitel mit einer kurzen Zusammenfassung der wichtigsten Ergebnisse, Erkenntnisse und Forschungsleistungen.

KAPITEL 2

KLINISCH-THERAPEUTISCHE PROZESSE

Dieses Kapitel stellt das für die Anästhesiologie sowie das für das Konzept des klinisch-therapeutischen Prozesses notwendige Hintergrundwissen bereit. Das Ziel ist es, weitere Grundlagen zum Verständnis der vorliegenden Problemstellung zu vermitteln.

Nach der Definition der Anästhesiologie und ihrer Teildisziplinen präzisieren Erläuterungen zur evidenzbasierten Medizin und den dort etablierten medizinischen Leitlinien mitsamt ihren Ausprägungen den weiteren Kontext für die in dieser Arbeit angewandte Methodik.

Mit der allgemeinen Begriffsbestimmung eines Prozesses sowie der Konkretisierung auf klinisch-therapeutische Anwendungsgebiete kann die wichtige Abgrenzung zu bereits etablierten Konzepten und vorhandenen Arbeiten erfolgen. In erster Linie die Besonderheit der klinisch-therapeutischen Prozesse in Verbindung mit Medizinprodukten bedarf der genaueren Betrachtung. Praxisnahe Beispiele für klinisch-therapeutische Prozesse verdeutlichen dies.

In der Medizin findet sich häufig ein generischer, inhärenter Behandlungszyklus, der dem Deming-Kreis ähnelt und den wir für klinisch-therapeutische Prozesse adaptieren. Dessen Kenntnis ist wichtig für die wissensbasierten Aspekte dieser Arbeit. Er wird demzufolge näher erläutert.

Das Kapitel schließt mit jeweils einem konkreten Beispiel für einen allgemeinen Prozess und einen klinisch-therapeutischen Prozess. Letzterer ist dem offiziellen Fundus der medizinischen Leitlinien entnommen.

2.1 ANÄSTHESIOLOGIE

2.1.1 BEGRIFFSBESTIMMUNG

Die Anästhesiologie ist ein hoch interdisziplinäres, medizinisches Fachgebiet, das nahezu sämtliche diagnostischen und therapeutischen Tätigkeiten des pflegerischen und ärztlichen Personals im klinischen Umfeld umfasst. Sie besteht aus den von Pasch et al. beschriebenen vier Teildisziplinen Anästhesie, Intensivmedizin, Notfallmedizin und Schmerztherapie [99], siehe Abbildung 2.1. Neuerdings wird auch die Palliativmedizin - also die Behandlung, Pflege und Begleitung von Patienten mit unheilbaren, fortgeschrittenen Erkrankungen und ihren Angehörigen - sowie das medizinische Management der Anästhesiologie zugeschrieben.

Allen Teildisziplinen gemein sind die stets wiederkehrenden Aktivitäten Therapie, Überwachung (Monitoring), Pflege und Dokumentation. Diese zentralen Aktivitäten mitsamt deren dabei verwendeter Medizintechnik sind wiederum - als klinische Prozesse formuliert - potenzielle Anwendungsfälle für das hier vorgestellte Software-Framework (SWFW).

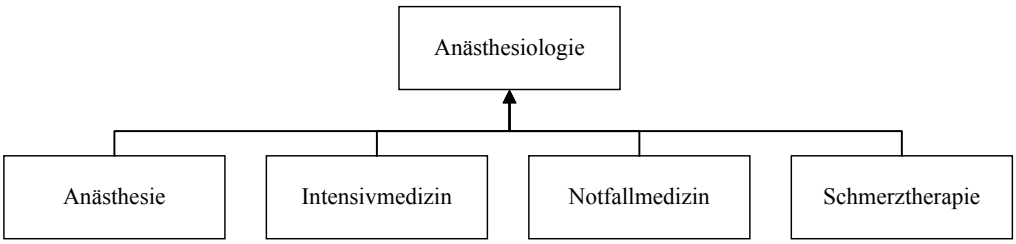

ABBILDUNG 2.1 – *Teildisziplinen der Anästhesiologie*

Die Teildisziplinen im einzelnen sind:

Die **Anästhesie** (Empfindungslosigkeit) als das klassische Feld der Anästhesiologie beschäftigt sich mit der reversiblen, umfassenden Lähmung des Zentralnervensystems (Narkose) von Patienten. Dazu zählt deren Bewusstlosigkeit (Hypnose), Schmerzfreiheit (Analgesie), Dämpfung vegetativer Reize sowie Muskelerschlaffung (Relaxation) zum Zwecke operativer Eingriffe.

Typische Medizintechnik: Anästhesiegerät, Patientenmonitor, Herz-Lungen-Maschine, Patientendaten-Managementsystem (PDMS).

Die **Intensivmedizin** behandelt kritisch und/oder lebensbedrohlich erkrankte Patienten mit kontinuierlicher Intensivpflege, Therapie und Überwachung. Hauptaufgabe der Intensivmedizin ist die Erhaltung und Wiederherstellung gestörter oder ausgefallener lebenswichtiger Organfunktionen. Patienten in diesem Akutpflegebereich verweilen entweder nur kurz (z. B. nach operativem Eingriff, Notfall) oder bis zu Tagen oder gar Monaten auf der Intensivstation. Spezielle Ausprägungen der Intensivmedizin mit entsprechenden Einrichtungen sind die Neonatologie (Neugeborene) und die Pädiatrie (Kleinkinder).

Typische Medizintechnik: Beatmungsgerät, Lungenmonitoring, Patientenmonitor, PDMS, Infusionspumpen, Dialysegeräte.

Die **Notfallmedizin** kümmert sich um alle akut erkrankten Menschen (medizinische Notfälle), wozu Erstversorgung, Rettungskette, Reanimation, Leitsymptomdiagnose, Monitoring, Transport, Notaufnahme sowie schließlich innerklinische Weiterversorgung gehören.

Die **Schmerztherapie** hilft chronisch schmerzkranken Menschen in dafür spezialisierten medizinischen Einrichtungen und stellt eine fachübergreifende Behandlung, die vor allem soziale und psychologische Faktoren berücksichtigt.

In dieser Arbeit liegt der Schwerpunkt auf den Teildisziplinen Anästhesie und Intensivmedizin. Jedoch können sämtliche hier vorgestellten Methoden, Prinzipien und Werkzeuge des Software-Frameworks auch auf klinische Prozesse der anderen Teildisziplinen angewandt werden.

Nachfolgend stellen wir wichtige Begriffe aus der Medizin vor, die eine zentrale Ausgangsbasis für die Entwicklung eines generativen Software-Framework sind.

2.1.2 EVIDENZBASIERTE MEDIZIN

Mit der evidenzbasierten Medizin (EBM) ist ein Wissenschaftsbereich entstanden, der die Qualität und Effektivität von Diagnose und Therapie systematisch, quantifizierbar und nachhaltig verbessern soll. Es geht darum, die ärztlichen Entscheidungsgrundlagen nicht allein auf individuelle und allgemeine Erfahrungen beruhen zu lassen, sondern diese mehr und mehr durch empirisch ermitteltes und validiertes Wissen zu ersetzen. Neben der individuellen klinischen Erfahrung bilden Werte und Wünsche der Patienten sowie die klinische Forschung die basalen Quellen für die Aktivitäten der EBM.

Das Cochrane-Institut [114] definiert auf ihrer Internet-Seite EBM wie folgt:

Definition EVIDENZBASIERTE MEDIZIN

> Evidenzbasierte Medizin ist der gewissenhafte, ausdrückliche und vernünftige Gebrauch der gegenwärtig besten externen, wissenschaftlichen Evidenz für Entscheidungen in der medizinischen Versorgung individueller Patienten. Die Praxis der EBM bedeutet die Integration individueller klinischer Expertise mit der bestverfügbaren externen Evidenz aus systematischer Forschung.

Damit bietet die EBM für die Anästhesiologie einen immensen Fundus an potenziellen Anwendungsfällen, wenn es darum geht, klinisch-therapeutische Prozesse per Software-Anwendungen digital zu unterstützen. Wissenschaftliche Evidenz und damit hohe Akzeptanz ist von entscheidender Wichtigkeit für Zulassung, Inverkehrbringung und Vermarktung von Medizinprodukten, die dieses Wissen umsetzen.

Übrigens fand im Jahre 1997 der erste EBM-Workshop des EBM-Netzwerkes am Institut für Sozialmedizin des Universitätsklinikums Schleswig-Holstein in Lübeck statt [92].

2.1.3 MEDIZINISCHE LEITLINIEN

Mit dem Konzept der in der Einleitung vorgestellten medizinischen Leitlinien nähern wir uns nun den prozessualen Anteilen der Anästhesiologie. Medizinische Leitlinien sind ein direktes, fundamentales und praktisch zu nutzendes Ergebnis der EBM bzw. den daran arbeitenden Instituten. Hier ist vor allem die Arbeitsgemeinschaft der Wissenschaftlichen Medizinischen Fachgesellschaften (AWMF) zu nennen, die in ihrem Portal eine Vielzahl qualitätsgeprüfter, medizinischer Leitlinien bündelt und der Allgemeinheit zur Verfügung stellt [55].

Ebenfalls im AWMF-Regelwerk [60] festgelegt ist das sogenannte S-Klassifikationsschema, mit dem für eine medizinische Leitlinie ein Grad der Evidenz als Qualitätsstufe definierbar wird, siehe Tabelle 2.1.

Klasse	**Bezeichnung**	**Methodiken**
S1	Handlungsempfehlungen von Expertengruppen	Konsensfindung in einem informellem Verfahren
S2k	Konsensbasierte Leitlinie	Repräsentatives Gremium, Strukturierte Konsensfindung
S2e	Evidenzbasierte Leitlinie	Systematische Recherche, Auswahl, Bewertung der Literatur
S3	Evidenz- und konsensbasierte Leitlinie	Repräsentatives Gremium, Systematische Recherche, Auswahl, Bewertung der Literatur, Strukturierte Konsensfindung

TABELLE 2.1 – *Stufenklassifikation medizinischer Leitlinien*

Evidenz, Konsens und Systematik steigt von S1- zu S3-Leitlinien. Je geringer die Evidenz einer Leitlinie, desto höher der Aufwand für Evaluierung und Marktzulassung daraus entwickelter Medizinprodukte.

Obwohl Sx-Leitlinien einen wertvollen Beitrag zur Qualitätssicherung und Wirksamkeit ärztlicher Tätigkeiten bieten, sind sie doch für die unmittelbare Extraktion eines modellierbaren, klinischen Prozesses zunächst eher ungeeignet. Gründe sind Format und Inhalte. So sind Leitlinien mehrheitlich textuell verfasst, sehr ausführlich und mit weitaus mehr Zusatzinformationen ausgestattet als die reinen Handlungsanweisungen zu Diagnose und Therapie. Es wird also Aufgabe des Software-Entwicklers sein, aus dedizierten Leitlinien geeignetes Prozesswissen als Design-Input für die Entwicklung von Medizinprodukten zu extrahieren.

Als exponiertes Beispiel sei hier die *S3-Leitlinie Sepsis - Prävention, Diagnose, Therapie und Nachsorge* angeführt, die in ihrer aktuellen Fassung (2020) von Brunkhorst et al. beschrieben ist [68]. Diese Leitlinie adressiert ein kritisches medizinisches Problem, nämlich die möglichst frühe Erkennung und das effektive Behandeln septischer Erkrankungen. Sepsis - oder im Volksmund auch Blutvergiftung genannt - bezeichnet einen akut lebensbedrohlichen Zustand des Patienten, der durch Intoxikation mit Mikroorganismen entsteht. Die Chancen,

eine septische Erkrankung zu überleben, schwinden dramatisch mit der Zeit, die ihre Erkennung erfordert. Jede Stunde Verzögerung des Therapiebeginns nach gesicherter Erkennung einer Sepsis senkt die Überlebenschance des Patienten um 7,6 % [98]. Das zeigt, wie überaus wichtig eine kontinuierliche, engmaschige Überwachung der relevanten Sepsisparameter ist. Definition und Epidemiologie der Sepsis finden sich im Artikel von Fleischmann-Struzek und Kollegen [59].

Die S3-Leitlinie zur Sepsis umfasst 73 Seiten Text und gliedert sich in 105 Empfehlungen samt Begründungen zu 23 Themenbereichen, siehe Tabelle 2.2. In Klammern die Anzahl Empfehlungen je Thema.

Prävention der Sepsis (8)	Impfungen (1)
Initiale hämodynamische Stabilisierung (8)	Leitlinienimplementierung (2)
Diagnose (2)	Antimikrobielle Therapie (14)
Fokuskontrolle (2)	Flüssigkeitstherapie (5)
Vasoaktive Medikation (9)	Kortikosteroide (1)
Blutprodukte (4)	Immunglobuline (1)
Blutreinigung (1)	Antikoagulanzien (1)
Invasive Beatmung (20)	Sedierung und Analgesie (1)
Blutzuckerkontrolle (3)	Nierenersatztherapie (3)
Bikarbonattherapie (1)	Stressulkusprophylaxe (3)
Ernährung (9)	Setzen von Behandlungszielen (2)
Prophylaxe einer venösen Thromboembolie (4)	

Tabelle 2.2 – *Themenbereiche der S3-Leitlinie zum Sepsismanagement*

Beispiel für eine dieser Empfehlungen aus dem Themenbereich *Diagnose*:

> Wir empfehlen, dass die Verabreichung von intravenösen Antiinfektiva so schnell wie möglich, idealerweise innerhalb einer Stunde, nach der Diagnose einer Sepsis oder eines septischen Schocks erfolgt.
> - SSC-Leitlinienadaptation
> - Konsensstärke: 100
> - Empfehlungsgrad: stark
> - Evidenzgrad: moderat

Diese S3-Leitlinie ist weitverbreitet im klinischen Alltag und wird uns während der gesamten Arbeit als Anwendungsbeispiel dienen.

2.1.4 LEITLINIENARTEN

Die Erkenntnis, mit einer qualitätsgesicherten Evidenz die Medizin nachhaltig zu verbessern, hat sich mittlerweile in vielen Ländern der Erde durchgesetzt. Gerade im amerikanischen Raum haben sich medizinische Leitlinien ebenfalls seit langem etabliert. Allerdings sind dort eine Reihe von Ausprägungen entstanden, die zwar im Kern derselben Zielsetzung folgen, evidenzbasierte Handlungsempfehlungen zu liefern, die sich jedoch in einigen Aspekten deutlich voneinander unterscheiden.

Diese Unterschiede sind enorm wichtig für die Identifikation klinischer Prozesse als geeignete Wissensquellen für eine Produktentwicklung.

Gerade weil medizinische Leitlinien Handlungsempfehlungen sind, spielt der Aspekt der Harmonisierung eine wichtige Rolle. Letztendlich steht alleinig der Arzt und das Pflegepersonal in Pflicht und Verantwortung für eine ethisch, moralisch einwandfreie Heilung seiner Patienten. Das gilt auch für den Fall, dass sie medizinische Leitlinien nutzen, unabhängig von deren Inhalt und Format. Für die Computerisierung medizinischer Leitlinien trifft dies ebenfalls zu, denn bei dieser Transformation wird es unausweichlich zu Inhaltsverlusten und Abweichungen kommen. Hersteller oder andere Produzenten dagegen können die Bedürfnisse ihrer Kunden aufnehmen und sogar mehrere medizinische Leitlinien in einem Medizinprodukt unterbringen. Auch in diesem Falle hat das Klinikpersonal die absolute und finale Entscheidungshoheit.

In Abschnitt 1.2 haben wir das Konzept der medizinischen Leitlinien allgemein eingeführt. Nachfolgend stellen wir nun unterschiedliche Ausprägungen und Präzisierungen medizinischer Leitlinien vor.

Eine *Standard Operating Procedure (SOP)*, zu deutsch etwa: Standardvorgehensweise, ist ein textueller Leitfaden, in dem Abläufe und Handlungsanweisungen für die Ausführung bestimmter Aufgaben oder Prozesse in einem Unternehmen oder einer Organisation detailliert festlegt sind. Es können auch Checklisten, Fragebögen und andere Hilfsmittel zum Einsatz kommen. SOP sind häufig eher allgemein und wenig spezifisch formuliert.

In der Anästhesiologie sind SOP weit verbreitet und fester Bestandteil der EBM-Bemühungen. Als Beispiel aus der Intensivmedizin sei hier die SOP zu „Beatmung und Extubation" genannt [48].

Inhaltlich und intentional einer Standard Operating Procedure sehr ähnlich ist die *Clinical Guideline (CG)*, auch Clinical Practice Guideline, zu deutsch: Klinische (Praxis) Leitlinie.

Definition CLINICAL GUIDELINE

> Klinische Leitlinien sind systematisch entwickelte Darstellungen und Empfehlungen mit dem Zweck, Ärzte und Patienten bei der Entscheidung über angemessene Maßnahmen der Krankenversorgung (Prävention, Diagnostik, Therapie und Nachsorge) unter spezifischen medizinischen Umständen zu unterstützen [102].

Klinische Leitlinien sind und werden gerade in den USA, aber auch in Europa, umfangreich erforscht. Zahlreiche Forschungsinstitute beschäftigen sich mit Vorteilen, Qualitätssicherung, Evidenz, Verfügbarmachung, Werkzeugunterstützung etc. [22] Neben den in Kapitel 1 gelisteten Vorzügen klinischer Leitlinien, hat Chatburn [95] darüber hinaus noch folgende Vorteile identifizieren können:

- Striktere Einhaltung evidenzbasierter Interventionen
- Reduktion therapeutischer Variabilitäten

- Verbesserter Patientendurchsatz
- Verbesserte Patientensicherheit
- Effizientere Aus- und Weiterbildung

Der Begriff des *Clinical Protocols* - zu deutsch etwa: Klinischer Plan - wird in zwei recht unterschiedlichen Interpretationen verwandt.

Von Chatburn [95] stammt die

Definition CLINICAL PROTOCOL

> Ein präziser und detaillierter Plan für die Untersuchung eines medizinischen Problems oder für ein Therapieschema.

In diesem Sinne meint ein klinischer Plan also die Zusammenstellung aller für die Durchführung einer klinischen Studie notwendigen Informationen in einem Planungsdokument, auch Studiendesign genannt. Als solcher ist er für unser Vorhaben, klinische Prozesse zu identifizieren, weniger relevant.

Dagegen wird im therapeutischen Sinne der Begriff *Clinical Protocol* häufig auch benutzt, um mehr spezifischere und detailliertere Handlungsanweisungen für diagnostische und therapeutische Lösungen zu standardisieren. Hierzu gehören Empfehlungen für Einstellungen und Verwendung von Messwerten medizinischer Geräte, die häufig schon in Form von Entscheidungsbäumen und/oder -diagrammen repräsentiert werden.

Jouvet et al. [90] definieren diese Interpretation eines klinischen Plans so:

Definition CLINICAL PROTOCOL (TECHNISCH)

> Ein klinischer Plan ist ein Dokument, das als Entscheidungshilfe für die Diagnose, das Management und die Behandlung bestimmter medizinischer Situationen dient.

Damit stellt sich ein Clinical Protocol als Spezialisierung eines Clinical Guideline dar und ist somit deutlich geeigneter als Design-Input für die Entwicklung von Software-Anwendungen für die Anästhesiologie.

Ein *Clinical Pathway* schließlich - zu deutsch etwa: klinischer Behandlungspfad - fokussiert mit seinen Inhalten und Repräsentationsformaten vornehmlich die prozessualen Aspekte standardisierter Handlungsabläufe. Rotter definiert wie folgt [50]:

Definition CLINICAL PATHWAY

> Klinische Behandlungspfade sind Instrumente, die der evidenzbasierten Gesundheitsversorgung dienen. Ihr Ziel ist es, Empfehlungen für klinische Praxisleitlinien in klinische Versorgungsprozesse innerhalb der spezifischen Kultur und Umgebung einer Gesundheitseinrichtung umzusetzen.

Klinische Behandlungspfade präzisieren die Vorgehensweise bei dedizierten ärztlichen Fragestellungen wie zum Beispiel „Wie entwöhne ich einen Patienten von der Beatmung?". Dabei nutzen sie ebenfalls grafische Repräsentationsformate, ergänzt mit zeitlichen Angaben zum Behandlungsablauf. Folglich sind klinische Behandlungspfade eine weitere Spezialisierung von Clinical Guidelines und Clinical Protocols mit einem prozessualen Schwerpunkt. Sie eignen sich damit vorzüglich als Design-Input für unser Software-Framework.

Zu beachten ist allerdings, dass es rechtliche Unterschiede bei der Verwendung o.g. Leitlinienausprägungen gibt. Es gilt: je spezifischer und individueller ärztliche Handlungsempfehlungen sind, desto höher das Haftungsrisiko im Falle eines Patientenschadens. Am allgemeinsten ist die klinische Leitlinie, am spezifischsten der klinische Behandlungspfad.

Fazit: alle o. g. Ausprägungen medizinischer Leitlinien bieten geeignetes klinisches Wissen für die Nutzung in einem Medizinprodukt. Sie unterscheiden sich durch Zielsetzung und -gruppe, Fokus, Umfang, Spezifität, Präzision, Repräsentationsformat und rechtlichen Aspekten. Letztendlich bleibt es des Software-Entwicklers Herausforderung, geleitet von den jeweiligen Produktanforderungen, das benötigte prozessuale Wissen zu extrahieren und für eine Computerisierung aufzubereiten. Im Kapitel 4 werden wir diese Aktivitäten genauer untersuchen.

2.2 PROZESSE

Mit der evidenzbasierten Medizin und dort speziell den medizinischen Leitlinien - idealerweise in Form klinischer Behandlungspfade - steht uns eine umfassende Quelle Expertenwissen für die Entwicklung medizinischer Software zur Verfügung. Betrachten wir nun die prozessualen Komponenten dieses Wissens, die uns als Ausgangsbasis für die Produktentwicklung dienen sollen.

Die Autoren der ISO 9000 [112] sprechen von einem Prozess als „einen Satz von in Wechselbeziehung oder Wechselwirkung stehenden Tätigkeiten, der Eingaben in Ergebnisse umwandelt".

Konkreter ist die Definition des Gabler-Wirtschaftslexikons [117]:

Definition PROZESS

> Ein Prozess ist eine Abfolge von Ereignissen, Zustandsänderungen oder Aktivitäten, die dazu dienen, ein bestimmtes Ziel oder Ergebnis zu erreichen. Dieses Konzept kann auf verschiedene Arten von Systemen angewendet werden, einschließlich natürlichen Phänomenen, technologischen Systemen, organisatorischen Strukturen und algorithmischen Prozeduren.

Für eine vollständige Prozessspezifikation und als Eingangsgröße für eine Software-Entwicklung braucht es folgende Angaben:

- Zielsetzung des Prozesses und erwarteter Mehrwert
- Eingänge und Ausgänge (Daten, Benutzereingaben, Ereignisse etc.)
- Rollen und Akteure, die am Prozess beteiligt sind

- Start-, Ende- und Zwischenereignisse
- Schnittstellen zu anderen Prozessen
- Prozessschritte (Aktivitäten) und die sie verbindende Logik

2.2.1 DER KLINISCHE BEHANDLUNGSZYKLUS

Seit den 50-er Jahren ist im Bereich Qualitätsmanagement eine Problemlösetechnik bekannt, die nach ihrem Erfinder Walter Deming als *Deming-Kreis* oder geläufiger *PDCA-Zyklus* (Plan-Do-Check-Act) - zu deutsch: Planen-Umsetzen-Überprüfen-Handeln - benannt ist [106]. Mit Aufkommen der agilen Software-Entwicklung hat der PDCA-Zyklus beachtliche Verbreitung erfahren und sich auch in anderen Bereichen der Wissenschaft und Industriebranchen etabliert.

Es handelt sich dabei um einen iterativen, vierphasigen Prozess, der das Lernen und die Verbesserung im Sinne eines kontinuierlichen Verbesserungsprozesses standardisieren soll. Inspiriert vom PDCA-Zyklus definieren wir einen generischen Behandlungszyklus für die Medizin - genauer gesagt für klinisch-therapeutische Prozesse, siehe Abbildung 2.2. Damit erhalten wir ein wiederverwendbares, invariantes Prozessmuster, bestehend aus den Prozessschritten Befund-Diagnose-Ziel-Therapie (BDZT). Dieses Prozessmuster, das wir nachfolgend als BDZT-Zyklus bezeichnen, ist allen klinisch-therapeutischen Prozessen enthalten.

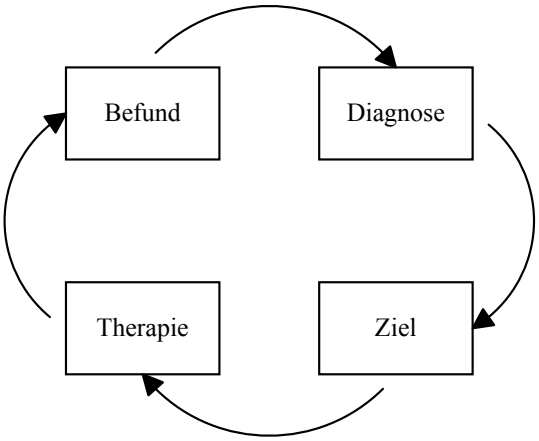

ABBILDUNG 2.2 – *Generischer Behandlungszyklus für klinisch-therapeutische Prozesse*

Die vier Prozessschritte im Einzelnen sind:

Befund. Vergleichbar einer Visite am Patientenbett. Anhand von Mess- und Einstellwerten der Medizingeräte, dem visuellen Eindruck vom Patienten, dokumentierter Patientendaten und weiteren, verfügbaren Informationen erfassen die behandelnden Personen die Ist-Situation und „machen sich ein Bild vom Patienten".

Diagnose. Aus diesem „Patientenbild" werden einzelne oder mehrere Diagnosen geschlussfolgert und vorherige gegebenenfalls korrigiert.

Ziel. Ein spezifisches Therapieziel wird abgeleitet, ein vorheriges ggfs. korrigiert und/oder angepasst.

Therapie. Maßnahmen - auch Interventionen genannt - zur Erreichung des Therapieziels werden durchgeführt. Dies können insbesondere Einstellungsveränderungen an den Medizingeräten und/oder Hinweise und Empfehlungen an den Anwender sein.

Der PDCA-Zyklus soll theoretisch nie enden, wohingegen der BDZT-Zyklus für gewöhnlich repetitiv während der Behandlung eines Patienten aktiv ist, getaktet beispielsweise durch Patientenvisiten und/oder etwaige Notfallmaßnahmen.

2.2.2 KLINISCH-THERAPEUTISCHE PROZESSE

In dieser Arbeit wird ein Software-Framework erarbeitet, mit dem klinisch-therapeutische Prozesse (KTP) computerisiert und von vielen Zielsystemen ausgeführt werden können. Mit obiger Prozessdefinition wird nun eine weitere Präzisierung des Begriffes speziell für klinisch-therapeutische Prozesse möglich. Es sind dies Prozesse im klinischen Umfeld, genauer: in der Anästhesiologie, die dadurch gekennzeichnet sind, dass sie

- eine spezifische Behandlung vollständig beschreiben
- eine definierte, therapeutische Zielstellung verfolgen
- auf der Intensivstation und/oder im Operationssaal angewendet werden
- mit Medizingeräten und/oder medizinischer Software interagieren
- mittel- und unmittelbar auf den Patienten einwirken
- dem in Abschnitt 2.2.1 vorgestellten Behandlungszyklus folgen

Ebenfalls wichtig ist die Abgrenzung klinisch-therapeutischer Prozesse von den sogenannten klinisch-administrativen Prozessen. Letztere sind Prozesse mit Patientenbezug aber ohne direkte Zuordnung zum Heilungsprozess, wie beispielsweise Leistungsabrechnung, Patientenaufnahme, Behandlungsplanung etc. [52], die damit in die Zuständigkeit der Krankenhaus-IT, dem Krankenhausinformationssystem, fallen. Einige kommerzielle Software-Hersteller bieten mittlerweile Produkte an, die angeben, klinische Prozesse computerbasiert zu unterstützen. Bei genauerer Betrachtung handelt es sich jedoch nicht um klinisch-therapeutische Prozesse, sondern um klinisch-administrative.

In Tabelle 2.3 sind Beispiele für klinisch-therapeutische Prozesse in der Anästhesiologie aufgeführt, die wir im Rahmen dieser Arbeit computerisiert haben. Insbesondere auf das Thema „Sepsis" werden wir immer wieder konkret eingehen.

Wir führen an dieser Stelle zwei Beispiele für Prozesse ein, von denen das erste bewusst nicht aus dem Bereich der klinisch-therapeutischen Prozesse stammt und den Aspekt der Allgemeingültigkeit der Lösung unterstreicht. Das zweite Beispiel adressiert dann einen klinisch-therapeutischen Prozess, dessen Entwicklung wir von der medizinischen Leitlinie bis zu einer kommerziellen Software-Anwendung verfolgen. Beide sind initial zunächst als Programmablaufplan (PAP), auch: Flussdiagramm, grafisch modelliert und werden im weiteren Verlauf der Arbeit sukzessive in jeweils relevante Repräsentationsformate überführt. Wir begegnen

Thema	Zielstellung	Teildisziplin	Akteure
Sepsis	Überwachung und Management	Intensivmedizin	Arzt, Pflege, PDMS
Glukose	Überwachung und Entscheidungs-unterstützung	Intensivmedizin	Arzt, Pflege, PDMS
Beatmung	Entwöhnung von der Beatmung	Intensivmedizin	Arzt, Pflege, Beatmungsgerät
Beatmung	Beatmung während der OP	Anästhesie	Arzt, Pflege, Anästhesiegerät
Delir	Überwachung und Management	Intensivmedizin Anästhesie	Arzt, Pflege, Patientenmonitor PDMS

Tabelle 2.3 – *Beispiele für klinisch-therapeutische Prozesse*

dem allgemeinen Beispiel in Kapitel 8 und dem klinisch-therapeutischen in Kapitel 5 und in Kapitel 8 wieder.

2.2.3 BEISPIELPROZESS, ALLGEMEIN

Das allgemeine Beispiel zeigt in Abbildung 2.3 den Prozess für die Entwicklung medizinischer Software gemäß der IEC 62304 [110] auf oberste Ebene, also als Top-Level-Diagramm, im Format eines Programmablaufplans (PAP). Wir haben dieses Beispiel ausgewählt, weil das hier vorgestellte Software-Framework ebenfalls nach diesem Prozess entwickelt wurde und typische Modellierungselemente Verwendung finden.

In der sich anschließenden Verfeinerung (Dekomposition) werden Aktivitäten als Subprozesse (Classify Software) oder separate Diagramme (z. B. Specify Software Requirements) weiter detailliert. Die Raute (Software approved?) symbolisiert logische Entscheidungspunkte mit entsprechenden Verzweigungen.

Inhaltlich lässt sich der Prozess wie folgt lesen: Unmittelbar nach dem Start des Entwicklungsprojektes wird die Software-Anwendung zunächst den Kriterien der IEC 62304 gemäß klassifiziert, damit im nächsten Prozessschritt die entsprechenden Anforderungen spezifiziert werden können. Danach entsteht die Software-Architektur, mit der die eigentliche Implementierung (Detail Software Design) eingeleitet wird. Sind dann alle Quelltexte erstellt worden, kann die Software kompiliert, integriert, getestet und freigegeben werden. Scheitert die Freigabe der Software, beispielsweise weil Tests wiederholt werden müssen oder Modifizierungen notwendig sind, so beginnt ein weiterer Entwicklungszyklus mit einer neuerlichen Spezifikationsphase.

2.2.4 BEISPIELPROZESS, KLINISCH-THERAPEUTISCH

Als Beispiel für die grafische Darstellung eines klinisch-therapeutischen Prozesses haben wir die bereits im Abschnitt 2.1.3 angeführte *S3-Leitlinie Sepsis - Prävention, Diagnose, Therapie und Nachsorge* [68] als Programmablaufplan in Abbildung 2.4 grafisch modelliert.

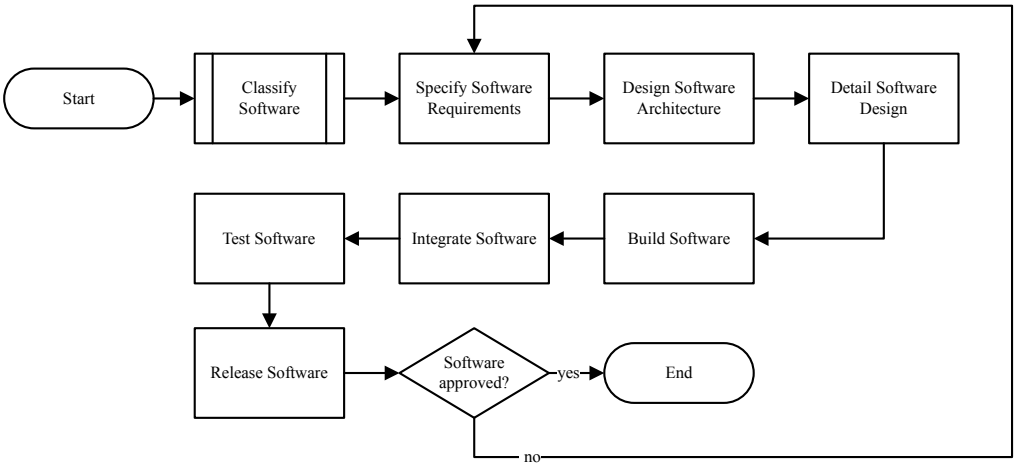

ABBILDUNG 2.3 – *Prozess „Design Software" als Programmablaufplan*

Wichtiges Detail in diesem Programmablaufplan ist neben dem Element der Teilprozesse (Rechtecksymbol mit 4 vertikalen Linien) vor allem die Parallelisierung von Prozessschritten (horizontales Doppelliniensymbol). Parallelisierung von Aktivitäten ist ein fundamentales Charakteristika von klinisch-therapeutischen Prozessen und damit eine zentrale Anforderung für die Implementierung des Software-Frameworks.

Die Erkennung und Behandlung septischer Erkrankungen ist darin wie folgt dargestellt: Nach Eingabe patientenspezifischer Daten wie Gewicht, Alter und Anamnese überwachen drei Subprozesse minütlich (*Wait 60sec*) in einer Art Eskalationsschema drei Sepsis-Krankheitsbilder.

Monitor for Sepsis. Dieser Subprozess empfängt gemäß der S3-Leitlinie festgelegte Parameter vom Zielsystem und prüft diese zyklisch auf Verletzung von Bereichsgrenzen. Ebenfalls für die Diagnosebildung herangezogen werden retrospektive Trends von Parametern und die Auswertung von zwischenzeitlich erfolgten Interventionen. Falls anhand der in der Wissensbasis hinterlegten Regeln eine Sepsis (Erste Eskalation) geschlussfolgert wurde, so ergeht an den Benutzer der Alarm „Sepsis detected !!!" und es starten zwei weitere Subprozesse, die pseudoparallel ausgeführt werden: *Handle Sepsis* und *Monitor for Severe Sepsis*.

Handle Sepsis. Handlungsempfehlungen zur Medikamentengabe, zum erweiterten Patientenmonitoring und anderen Maßnahmen, die die S3-Leitlinie vorgibt, werden via GUI an den Benutzer gegeben, um eine erkannte Sepsis zu behandeln.

Monitor for Severe Sepsis. Zweite Eskalation. Analog zu *Monitor for Sepsis*, inklusive der parallelen Teilprozesse *Handle Severe Sepsis* und *Monitor for Septic Shock*.

Handle Severe Sepsis. Analog zu *Handle Sepsis*.

Monitor for Septic Shock. Dritte Eskalation. Analog zu *Monitor for Severe Sepsis*.

Handle Septic Shock. Analog zu *Handle Severe Sepsis*.

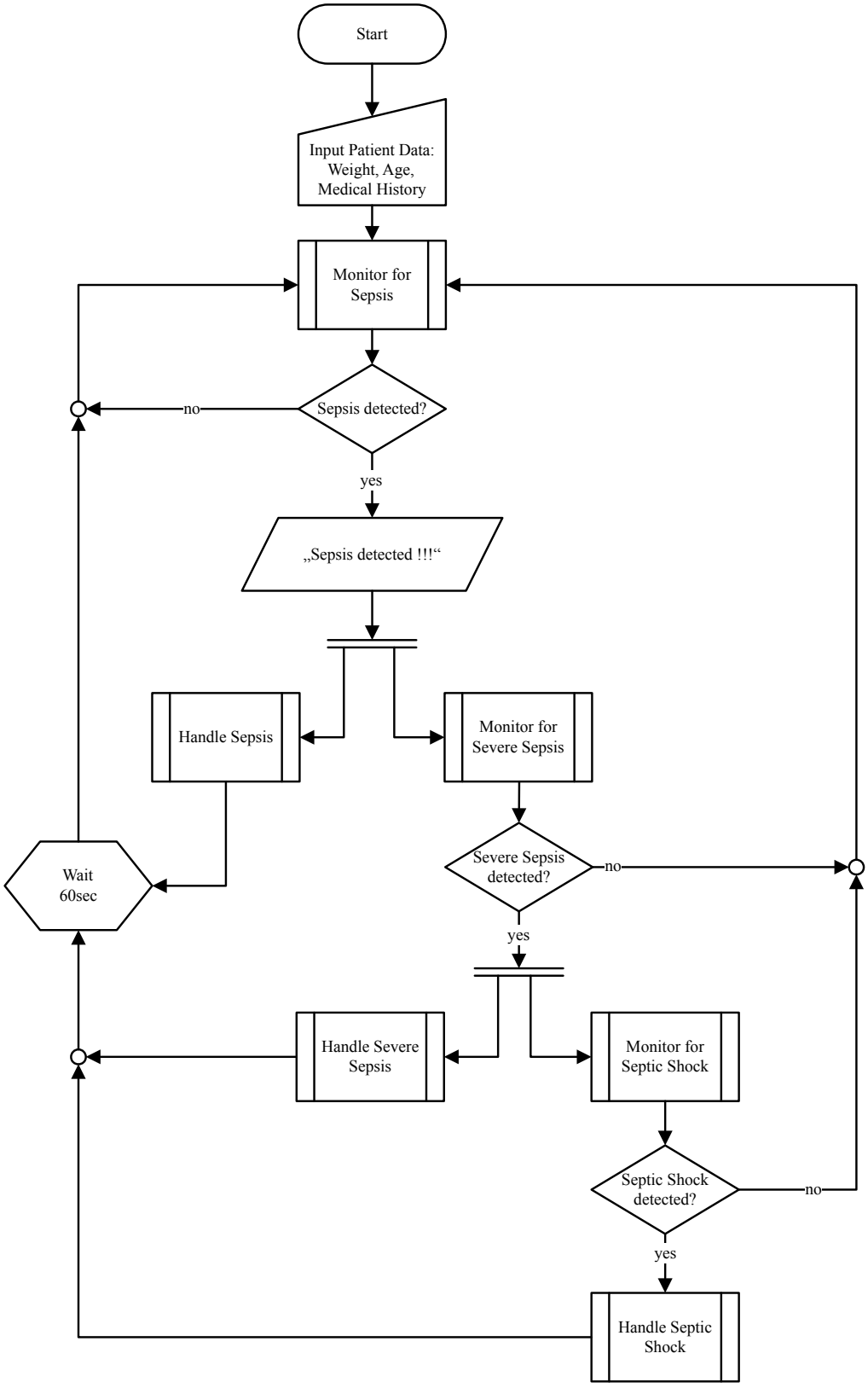

Abbildung 2.4 – *Prozess „Manage Sepsis" als Programmablaufplan*

Wir haben mit diesem Kapitel den praktischen Anwendungsbereich dieser Arbeit vorgestellt, sowie den medizinischen Kontext der evidenzbasierten Medizin und dort im Besonderen der medizinischen Leitlinien als Quelle für klinische Prozesse erläutert.

Der zentrale Begriff des Prozesses wurde definiert und auf das klinische Umfeld abgebildet. Neuartig im Vergleich zu verwandten Arbeiten und dem gegenwärtigen Stand in der Fachwelt ist die konkrete Spezialisierung klinischer Prozesse zu klinisch-therapeutischen Prozessen und deren präzise Abgrenzung zu klinisch-administrativen Prozessen, wie sie beispielsweise Swoboda definiert [52]. Die Einführung dieser Verfeinerung unterstreicht und festigt die Tatsache, dass es sowohl um der Patientenheilung dienliche Vorgänge geht, als auch dass eine mandatorische Interaktion mit mindestens einem Medizingerät stattfindet.

Mit dem generischen, klinischen Behandlungszyklus ist ein neues Charakteristikum geschaffen, dass die Identifikation klinisch-therapeutischer Prozesse vereinfacht. Zwei Prozessbeispiele dienen der Veranschaulichung und werden uns durchgängig begleiten, um die methodische Evolution des Lösungsweges zu verdeutlichen.

KAPITEL 3

WISSENSVERARBEITENDE SYSTEME

Im vorangehenden Kapitel wurden die Grundlagen geschaffen, welche den Anwendungsbereich des Software-Frameworks sowie das spezifische Konzept der klinisch-therapeutischen Prozesse betreffen.

Mit den wissensverarbeitenden Systemen stellen wir in diesem Kapitel die allgemeine Methodik vor, die wir für die Lösung der Problemstellung ausgewählt haben. Nach einer Bestimmung des Begriffes Wissen und der Einordnung dieser Thematik als Teilgebiet der künstlichen Intelligenz gehen wir näher auf Systematik und Architektur von Expertensystemen ein, bevor wir detailliert die Besonderheiten wissensbasierter Systeme als Sonderform der Expertensysteme beschreiben. Dazu zählt auch die Art und Form wie Wissen repräsentiert werden kann.

Die Problemlösungsstrategie wiederum ist die zentrale Logikeinheit eines wissensverarbeitenden Systems. Sie ermöglicht die Verknüpfung des Wissens durch logisches Schließen, um eine gegebene Fragestellung zu beantworten. Wir stellen verschiedene Typen von Problemlösungsstrategien vor, die der Lösungsfindung unserer Problemstellung zur Verfügung stehen und erläutern deren unterschiedliche Anwendungsfälle.

Eine besondere Spezialisierung wissensbasierter Systeme stellen semantische Wikis, auch Knowledge Wikis genannt, dar. Wir beleuchten die Prinzipien, Einsatzbereiche und basalen Eigenschaften von Knowledge Wikis und die Vorteile, die sie für die vorliegende Aufgabenstellung bieten.

3.1 WISSEN UND KÜNSTLICHE INTELLIGENZ

In einem klinisch-therapeutischen Prozess eingebettet ist das prozessuale Wissen zur Lösung eines spezifischen, medizinischen Problems. Ein tieferes Verständnis des Begriffes Wissen ist hilfreich für die Definition und Erarbeitung basaler Systemanforderungen und -eigenschaften, die wiederum inhärent in der Systemarchitektur umgesetzt sind.

Reimer [46] definiert in seinem Buch zu Wissensrepräsentationen den Begriff Wissen wie folgt

Definition WISSEN

> Als das Wissen eines Wissensträgers definieren wir die Menge aller von ihm als wahr angenommenen Aussagen über eine repräsentierte Welt, die tatsächlich wahr sind.

Das bedeutet mithin auch, dass zu den wahren Aussagen eines Wissensträgers auch die jeweiligen individuellen Überzeugungen hinzukommen, d. h. diejenigen Wissensinhalte, die der Wissensträger als wahr erachtet, die jedoch nicht notwendigerweise wahr sind. Also der Anteil Wissen, den wir - gerade vor dem Hintergrund der klinischen Evidenz, siehe Abschnitt 2.1.2 - besonders kritisch betrachten sollten.

Generell lässt sich feststellen: je individueller das mit einem klinisch-therapeutischen Prozess spezifizierte Wissen ist, desto höher werden Aufwand und Risiko für den Nachweis der Evidenz, da oftmals der Nachweis der Allgemeingültigkeit fehlt und/oder nicht ausreichend ist. Und je schwieriger ist es, Vorbehalte bei Anwendern abzubauen sowie Akzeptanz und Bereitschaft für die Nutzung einer Software-Anwendung zu schaffen, die klinisch-therapeutische Prozesse ausführt. Auf der anderen Seite bringt eine hohe klinische Evidenz häufig eine gewisse generalisierende Abstraktheit mit sich, die eine konkrete Umsetzung erschwert und die Hinzuziehung eben jenes individuellen Wissens erforderlich macht. Das trifft besonders für Parametereinstellungen von im klinisch-therapeutischen Prozess involvierter Medizintechnik zu. Es gilt, dieses vermeintliche Dilemma stets im Blick zu halten und kontinuierlich abzuwägen, welche Auswirkungen sich daraus für Entwicklungsprojekte ergeben könnten.

Das Konzept Wissen stellt für die künstliche Intelligenz (KI) das tragende Fundament dar, gehen doch sämtliche „intelligenten" Vorgänge mit der intensiven Nutzung und Verarbeitung von Wissen einher. Winston bezeichnet in seinem renommierten Werk [40] künstliche Intelligenz als das „Studium von Ideen, die Computer befähigen sollen, intelligent zu agieren". Problematisch hierbei ist, dass es bis dato keine universell etablierte Definition des Begriffes „Intelligenz" gibt. Technisch gesehen ist dies jedoch für diese Arbeit von untergeordnetem Interesse und so verstehen wir unter

Definition KÜNSTLICHER INTELLIGENZ

> Wissen nutzende Strategien, die die Anzahl möglicher Problemzustände optimal reduzieren, korrekte und nicht korrekte Problemlösungen identifizieren sowie irrelevante Lösungswege ausschließen, um eine spezifische Aufgabe ähnlich wie ein Mensch zu lösen.

3.2 EXPERTENSYSTEME

Klinisch-therapeutische Prozesse in der Anästhesiologie repräsentieren hoch spezialisiertes Wissen, sogenanntes Expertenwissen. Wird dieses Expertenwissen nach konventioneller Art und Weise in Software-Anwendungen eingebettet, so verschmilzt es mit Ablauf- und Steuerungslogik sowie technischen Details zu Algorithmen, die während ihrer Ausführung Daten verarbeiten, siehe Abbildung 3.1(a). Dadurch wird es dem jeweiligen Domänenexperten - hier: Ärzte, Pflegepersonal - nahezu unmöglich, ihr zuvor geliefertes Expertenwissen zu identifizieren, zu interpretieren und damit auch zu prüfen und ggfs. zu korrigieren bzw. zu pflegen.

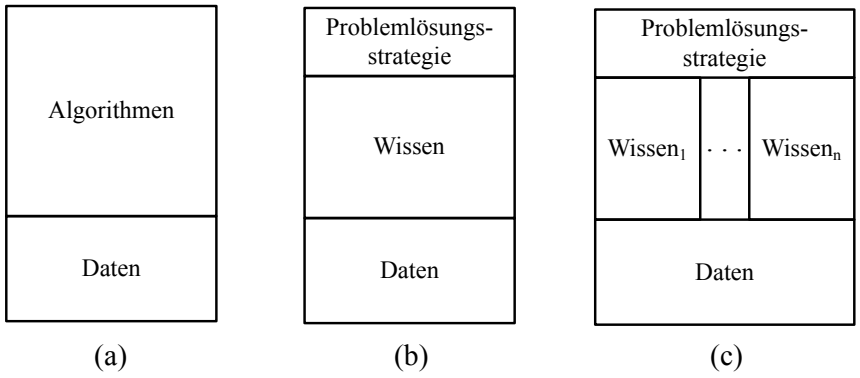

ABBILDUNG 3.1 – *(a) konventionell (b) Expertensysteme (c) multiple Wissensbasen*

Grundlegend anders ist dies bei Expertensystemen (XPS), die Puppe [29] folgendermaßen beschreibt:

Definition EXPERTENSYSTEM

> Expertensysteme sind Computerprogramme, mit denen das Spezialwissen und die Schlussfolgerungsfähigkeit qualifizierter Fachleute auf eng begrenzten Aufgabengebieten nachgebildet werden soll.

Hier ist das Expertenwissen explizit von der es interpretierenden Problemlösungsstrategie und den anliegenden Daten separiert, siehe Abbildung 3.1(b). Dieses systemische Prinzip bietet gleich mehrere signifikante Vorteile. So lassen sich einfach invariante Komponenten des Systems von variablen Komponenten trennen, was in vielerlei Hinsicht zu einem effizienteren Software-Lebenszyklus inklusive einer verbesserten Wartbarkeit führt. Die explizite Formalisierung des

Expertenwissens in einer Wissensbasis (WB) beschert dem Domänenexperten die kontinuierliche Interpretation, Prüfung und Weiterentwicklung seines Beitrags, also des klinisch-therapeutischen Prozesses, auch ohne computer-technisches Detailwissen. Lässt die Systemarchitektur dann auch noch den Betrieb mehrerer Wissensbasen mit gleicher Problemlösungsstrategie zu, zwischen denen während der Ausführung gewechselt werden kann, so können - wie in Abschnitt 1.4 gefordert - auch mehrere klinisch-therapeutische Prozesse für ein Zielsystem implementiert werden, siehe Abbildung 3.1(c).

Puppe [29] stellt eine allgemeingültige Architektur für Expertensysteme vor (Abbildung 3.2) und fasst deren praktischen Nutzen bezogen auf die Medizin wie folgt zusammen:

- Verbesserungen in der Genauigkeit medizinischer Entscheidungen
- Sicherung eines Minimalstandards
- schnellere Verfügbarkeit von medizinischem Wissen für das behandelnde Personal
- Verbesserung der Kosteneffizienz medizinischer Untersuchungen
- Kontrolle medizinischer Leistungen

All dies sind Vorteile, die wir auch für die Zielsetzung des Software-Frameworks (SWFW) reklamieren. Bezogen auf Abbildung 3.1 finden wir die Problemlösungsstrategie in der *Problemlösungskomponente* wieder, wohingegen sich das Wissen auf die Komponenten *fallspezifisches Wissen* und *bereichsbezogenes Expertenwissen* aufteilt.

Für die Entscheidung, ob Expertensysteme für eine gegebene Aufgabenstellung Methodik und Technik der Wahl sind, ist es dienlich, sogenannte Problemlösungstypen heran zu ziehen, die dabei helfen, einen Anwendungsbereich zu typisieren. Etablierte Problemlösungstypen können sein: Diagnostik, Überwachung, Konstruktion, Planung, Vorhersage, Simulation.

Der Anwendungsbereich Anästhesiologie sowie der im Abschnitt 2.2.1 vorgestellte generische Behandlungszyklus Befund-Diagnose-Ziel-Therapie (BDZT) implizieren die Zuordnung der vorliegenden Aufgabenstellung in die Kategorie *Diagnostik*. Dies bedeutet, dass die Problemlösung für eine anwendungsspezifische Aufgabe durch Auswahl aus einer Menge vorgegebener Alternativen gefunden wird, deren Herleitung u. a. durch Merkmalswerte bestimmt wird.

Zu beachten ist dabei, dass in dieser allgemeingültigen Expertensystem-Architektur zwei Phasen der Nutzung solcher Systeme nicht explizit, sondern vielmehr kombiniert dargestellt sind. Es sind dies die Phase der *Modellierung* und die der *Ausführung*. Wir gehen im Kapitel 4 auf diesen Umstand ausführlich ein. Angewandt auf den medizinischen Systemkontext und für klinisch-therapeutische Prozesse ergeben sich aus Abbildung 3.2 folgende Szenarien, die sowohl für den Open-Loop- als auch für den Closed-Loop-Betrieb gelten, siehe Abschnitt 1.4.

Der *Benutzer* wird durch den Arzt und/oder das Pflegepersonal repräsentiert, ebenso wie der *Experte* (Domänenexperte). Alle weiteren Teile des XPS sind Software-Komponenten des Medizingerätes und stehen mit diesem in Interaktion. Die Anwendung des Expertensystems für einen bestimmten Patienten wird als

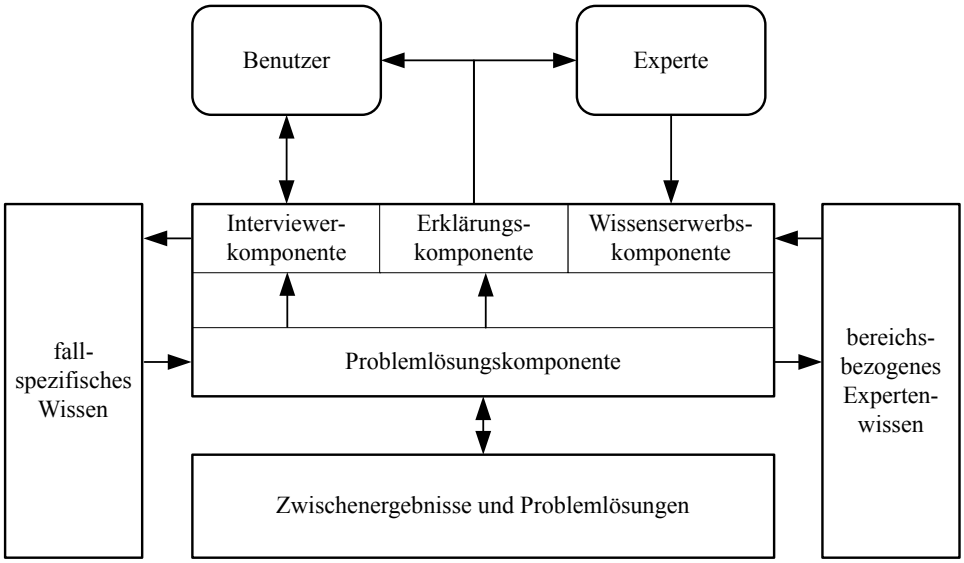

Abbildung 3.2 – *Allgemeine Architektur eines Expertensystems*

Fall (engl. Session) bezeichnet, die dabei entstehenden Laufzeitdaten werden als *fallspezifisches Wissen* (Ableitungs- und Steuerungswissen) abgelegt. Mit Hilfe der *Wissenserwerbskomponente* hat der Experte für die Entwicklung des XPS sein *bereichbezogenes Expertenwissen* (Faktenwissen) niedergelegt. Diese Komponente stellt zusammen mit dem fallspezifischen Wissen die Wissensbasis dar, mit der ein ausgewählter klinisch-therapeutischer Prozess (KTP) formal repräsentiert ist. Zu Beginn eines Falles und während dessen Ausführung gibt der Benutzer via Graphical User Interface (GUI) des Medizingerätes entsprechend den Vorgaben der Wissensbasis (i. e. klinisch-therapeutischer Prozess) über die *Interviewerkomponente* Daten ein. Während der Ausführung eines Falles werden zyklisch Daten vom Medizingerät und ggfs. vom Benutzer an die *Problemlösungskomponente* - auch Inferenzmaschine genannt - geliefert, die unter Nutzung der Wissensbasis *Zwischenergebnisse und Problemlösungen* ermittelt, die wiederum schrittweise der finalen Problemlösung entgegen streben. Dieser Vorgang wird als Inferenz (engl. Reasoning) bezeichnet.

Schlussendlich kann eine *Erklärungskomponente* den Benutzer/Experten typischerweise - aber nicht ausschließlich - über das GUI des Medizingerätes kontinuierlich über aktuelle Systemzustände, Zwischenergebnisse, erreichte Problemlösungen etc. informieren.

Wir werden dieses übergeordnete, abstrakte Funktionsprinzip in den kommenden Kapiteln weiter verfolgen, detaillieren und schließlich als zentralen Teil des Software-Frameworks implementieren.

Moderne Expertensysteme verfügen häufig zusätzlich zu der oben vorgestellten Grundarchitektur (Abbildung 3.2) noch über eine *Lernkomponente*, die es gestattet, während der Ausführung dynamisch Wissen anzupassen und so die Wissensbasis fortlaufend zu optimieren. Für die Zielsetzung dieser Arbeit ist solch

eine Dynamik jedoch nicht relevant, ja sogar eher unerwünscht.

Als Beispiele für prominente Expertensysteme seien hier zwei akademische Programme aus den Anfängen der Forschung angeführt. Beide sind an der Stanford Universität in den Vereinigten Staaten von Amerika entstanden und beschäftigten sich mit den Bereichen Medizin und Chemie.

MYCIN ist die PhD-Arbeit von Shortliffe [66] aus dem Jahre 1975 und hatte das Ziel, eine intelligente Entscheidungsunterstützung für die Diagnose und Therapie von bakteriellen Infektionskrankheiten, insbesondere bezüglich der Antibiotika-Gabe, zu liefern. Das System besticht durch eine methodische Klarheit, indem es eine konsequente Trennung von Wissensrepräsentation und Problemlösungsstrategie einhält. Die initiale Wissensbasis von MYCIN umfasst ca. 450 Regeln. MYCIN erreichte über die Jahre Vorbildcharakter für nachfolgende Generationen von Expertensystemen. Interessanterweise thematisierte die medizinische Problemstellung von MYCIN bereits zu jener Zeit den heute wieder und immer noch sehr prekären klinisch-therapeutischen Prozess des Managements septischer Erkrankungen, siehe Abschnitt 2.2.4.

DENDRAL wiederum entstand im Jahre 1978 aus dem Zusammenschluss einer großen, interdisziplinären Gruppe von Forschern der Stanford Universität unter der Federführung von Buchanan [58]. DENDRAL sollte vornehmlich dem Zweck dienen, Methoden der künstlichen Intelligenz zu explorieren. Potenzielle Anwender waren Chemiker, die in DENDRAL hinterlegtes Expertenwissen beispielsweise zur Identifikation und Analyse molekularer Strukturen via Massenspektrometrie nutzen sollten. Schon damals war die Rede von einem „Smart Assistant".

Beide Systeme fanden übrigens wenig bis keine nachhaltige Akzeptanz, da viele Anwender kaum die Bereitschaft zeigten, sich auf Problemlösungsvorschläge solch einer neuartigen Technologie zu verlassen.

3.3 WISSENSBASIERTE SYSTEME

Das dieser Arbeit übergeordnete Leitprinzip ist es, vom Programmieren zum Spezifizieren zu gelangen. In der Praxis bedeutet dies, gemeinsam mit Domänenexperten deren Wissen unkompliziert und effektiv zu erheben, zu formalisieren, zu sichten, zu validieren und weiter zu entwickeln. Gerade im Bereich der Medizin ist es von entscheidender Bedeutung, dass dieses kooperative Vorgehen möglichst einfach und frei von technischen Barrieren erfolgen kann. Wir haben bereits an mehreren Stellen herausgestellt, dass eine durchgehende Abgrenzung zwischen prozeduralen (Steuerungslogik) und deklarativen Anteilen (Wissensbasis) dieses Leitprinzip geeignet unterstützt.

Expertensysteme bieten qua Grundstruktur und -prinzip solch eine stringente Separierung und ermöglichen damit systemisch eine klare Trennung von invarianten und variablen Anteilen. Gerade die hochvariable Wissensbasis von den Steuerungsalgorithmen zu isolieren, hilft den Domänenexperten bei der kooperativen Entwicklung und Pflege der Software-Anwendung.

Wissensbasierte Systeme wiederum sind eine Spezialisierung der Expertensysteme und gehen bezüglich o. g. Trennung noch einen Schritt weiter, indem sie besonderen

Fokus auf die explizite Repräsentation des Wissens legen. Das hat zu einer Vielzahl von Wissensrepräsentationsformaten geführt, die im Wesentlichen das Ziel verfolgen, den gewählten Wissensbereich adäquat zu modellieren. Adäquat meint in diesem Falle einfach verständlich für den Domänenexperten, aber dennoch präzise und vollumfänglich, um den Wissensbereich verlustfrei abbilden zu können.

Definition WISSENSREPRÄSENTATION

> Von einer Repräsentation sprechen wir, wenn zusätzlich zu einer Menge von Repräsentationsstrukturen Angaben dazu vorliegen, wie die Strukturen der Repräsentation auf die Merkmale der repräsentierten Welt abzubilden sind. Diese Angaben stellen die Interpretationsvorschrift dar [46].

Die rein strukturelle Formalisierung eines gegebenen Wissensbereiches allein reicht folglich nicht aus. Erst die Interpretationsvorschrift ermöglicht eine dynamische Perspektive, mit der die Objektstrukturen, Merkmale und Beziehungen zwischen ihnen sichtbar werden. Zuvor haben wir die vorliegende Aufgabenstellung dem Problemlösungstyp Diagnostik zugeordnet. Für die zu wählende Interpretationsvorschrift muss also gelten, dass sie exakt diesen Problemlösungstyp realisieren kann, also die Auswahl einer Problemlösung aus einer Menge von Merkmalen und gegebenen Lösungsalternativen. Es handelt sich also um eine surjektive Funktion $i : M \rightarrow L$ einer Menge von Problemmerkmalen M und einer Menge von Problemlösungen L, wobei i die Interpretationsvorschrift ist.

Formal: $\forall m \in M \; \exists l \in L : f(m) = l$

Tabelle 3.1 listet alle Typen von Merkmalen und Lösungen für klinisch-therapeutische Prozesse.

Merkmale	**Lösungen**
Benutzereingaben	Diagnosemeldungen
Messwerte	Statusmeldungen
Ist-Einstellwerte	Soll-Einstellwerte
Alarmmeldungen	...
Statusmeldungen	

TABELLE 3.1 – *Generelle Eingangs- und Ausgangsgrößen eines klinisch-therapeutischen Prozesses*

Mit diesen Eingangs- und Ausgangsgrößen lässt sich nun der in Abschnitt 2.2.1 vorgestellte generische, klinische Behandlungszyklus Befund-Diagnose-Ziel-Therapie (BDZT) instanziieren. So wird aus allen Merkmalstypen eine Befundung (B) hergeleitet, während textuelle Diagnosemeldungen die schlussgefolgerten Diagnosen (D) anzeigen. Statusmeldungen identifizieren das avisierte, therapeutische Ziel (Z), das schließlich anhand ermittelter Soll-Einstellwerte sukzessive die Therapie (T) umsetzt. Messwerte, Einstellwerte sowie Alarm- und Statusmeldungen werden von den beteiligten Medizinprodukten geliefert.

Eine besondere Herausforderung stellt die Situation dar, dass der Problemlösungsvorgang, also die Anwendung der Interpretationsvorschrift mehrere Lösungen hervorbringt. Für den Bereich Diagnostik, speziell in der Medizin, sind

Mehrfachlösungen eher die Regel als die Ausnahme. In solch einem Fall ist die Kombination mehrerer Interpretationsvorschriften angezeigt, die den Umgang mit Mehrdeutigkeiten, Unsicherheiten, Unvollständigkeiten und Wahrscheinlichkeiten beherrschbar macht.

3.4 PROBLEMLÖSUNGSSTRATEGIEN

Synonym und geläufiger für den Begriff Interpretationsvorschrift ist die Bezeichnung *Problemlösungsstrategie*, die wir im Folgenden verwenden wollen. Mit Bezug auf Abschnitt 2.2 lässt sich feststellen, dass auch die Problemlösungsstrategie selbst ein Prozess ist. Das Ausführen dieses Prozesses auf eine Wissensbasis nennt sich *logisches Schließen*, engl. Reasoning.

Dem inhärenten Ziel, den generischen BDZT-Zyklus zu implementieren, haben wir uns im vorangehenden Abschnitt genähert, indem wir Eingangs- und Ausgangsgrößen für klinisch-therapeutische Prozesse identifiziert haben. Das logische Schließen versetzt nun den BDZT-Zyklus in Bewegung und verleiht ihm die notwendige Dynamik. Es gibt eine Fülle unterschiedlicher Problemlösungsstrategien, die spezialisiert für die jeweilige Domäne auf die Problemlösungstypen und andere, kontextsensitive Besonderheiten eingehen [44]. Für den hier im Fokus liegenden Problemlösungstyp Diagnostik existieren folgende sechs Problemlösungsstrategien, die auch miteinander kombiniert werden können.

Regelbasiertes Schließen ist eine Problemlösungsstrategie, die zur Ableitung von Schlussfolgerungen aus einer Wissensbasis verwendet wird, die aus einer Menge von Regeln besteht. Diese Methode ist besonders in Expertensystemen verbreitet, wo sie genutzt wird, um auf Basis eines gegebenen Regelwerks fundierte Entscheidungen zu treffen oder Probleme zu lösen. Ein regelbasiertes System besteht aus einer Menge von Regeln, der Regelbasis, einer Wissensbasis, die Fakten und Daten enthält, und einem Mechanismus zum logischen Schließen, der die Regeln anwendet, um neue Fakten oder Entscheidungen abzuleiten.

Es gibt zwei Inferenzmechanismen für das regelbasierte Schließen:

- Vorwärtsverkettung (Forward Chaining): Beginnt mit den verfügbaren Fakten und wendet Regeln an, um neue Fakten zu erzeugen, bis ein vorgegebenes Ziel erreicht wird.
- Rückwärtsverkettung (Backward Chaining): Beginnt mit einem zu beweisenden Ziel und arbeitet rückwärts, um festzustellen, welche Fakten und Regeln erforderlich sind, um dieses Ziel zu erreichen.

Regelbasierte Systeme werden häufig innerhalb von Expertensystemen eingesetzt, die eine Entscheidungsunterstützung in spezialisierten Bereichen wie Medizin, Recht und Ingenieurwesen bieten. Diese Systeme nutzen Regeln, um Diagnosen zu stellen oder rechtliche Empfehlungen auszusprechen.

Constraints definieren Randbedingungen, die von der Lösung eingehalten werden müssen. Sie sind entscheidend für die Modellierung und Lösung von Problemen in vielen Bereichen, da sie eine formale Möglichkeit bieten, Bedingungen zu spezifizieren und so die Lösungsfindung effizienter und gezielter zu gestalten. Constraints ermöglichen es, komplexe Probleme zu strukturieren und eine

Vielzahl von Lösungsansätzen zu evaluieren, um optimale oder zumindest zufriedenstellende Ergebnisse zu erzielen. Sie helfen also dabei, den Lösungsraum einzugrenzen und sicherzustellen, dass die gefundenen Lösungen bestimmten Anforderungen genügen. Im Gegensatz zu regelbasierten Ansätzen geben Constraints keine Ableitungsrichtung vor.

Probabilistisches Schließen ist eine Problemlösestrategie aus dem Bereich der Statistik, die genutzt wird, um Schlussfolgerungen aus unsicheren oder unvollständigen Informationen zu ziehen. Sie basiert auf der Wahrscheinlichkeitstheorie und dient dazu, berechnete Wahrscheinlichkeiten zuzuweisen und Entscheidungen unter Unsicherheit zu treffen.

Probabilistisches Schließen verwendet Konzepte wie bedingte Wahrscheinlichkeiten und Bayessche Netzwerke, um Zusammenhänge zwischen verschiedenen Variablen oder Ereignissen zu modellieren. Eine wichtige Komponente ist die bedingte Wahrscheinlichkeit, die beschreibt, wie wahrscheinlich ein Ereignis eintritt, wenn ein anderes bekanntes Ereignis zuvor bereits eintrat. Eine häufig verwendete Methode im probabilistischen Schließen ist das Bayessche Schließen. Es verwendet den *Bayesschen Satz*, um die Wahrscheinlichkeit einer Hypothese zu aktualisieren, nachdem neue Evidenz oder Daten beobachtet wurden. Probabilistisches Schließen wird in vielen Bereichen eingesetzt, darunter Diagnosesysteme, Vorhersagemodelle, maschinelles Lernen, natürliche Sprachverarbeitung und Entscheidungsunterstützungssysteme (engl. Decision Support Systems).

Die Methode ermöglicht es, trotz Unsicherheiten fundierte Entscheidungen zu treffen, und kann komplexe Systeme und deren Unwägbarkeiten modellieren. Durch die Anwendung probabilistischer Methoden können Systeme flexibler auf neue Informationen reagieren und dynamischere und präzisere Vorhersagen und Entscheidungen treffen. Durch probabilistisches Schließen können Modelle dynamischer gestaltet werden, sodass sie nicht nur auf festen Regeln basieren, sondern flexibel auf neue Informationen und Daten reagieren können, was die Prognose- und Entscheidungsfähigkeit in komplexen und unsicheren Umgebungen erheblich verbessert.

Nicht-monotones Schließen ist eine Problemlösungsstrategie aus der Logik, die sich mit der Fähigkeit befasst, Schlussfolgerungen zu revidieren oder zurückzuziehen, wenn neue Informationen hinzukommen. Im Gegensatz zu monotonen, logischen Systemen, in denen einmal getroffene Schlussfolgerungen stets gültig bleiben, ermöglicht nicht-monotones Schließen eine dynamische Anpassung der Wissensbasis an neue Daten oder geänderte Umstände.

In der klassischen, monotonen Logik führt das Hinzufügen neuer Erkenntnisse oder Prämissen nicht dazu, dass bestehende Schlussfolgerungen ungültig werden können. Nicht-monotone Logik erlaubt hingegen, dass zusätzliche Informationen bisherige Schlussfolgerungen ungültig machen und ersetzen können. Nicht-monotones Schließen ist besonders relevant in Systemen, die mit Unsicherheiten oder unvollständigen Informationen arbeiten, da es erlaubt, Schlussfolgerungen anzupassen, wenn sich der Informationsstand ändert, was in der Medizin unabdingbar ist.

Nicht-monotones Schließen ist in Expertensystemen geläufig, die in dynamischen

Umgebungen arbeiten und ihre Entscheidungen anpassen müssen, wenn sich die verfügbaren Informationen ändern. Durch die Fähigkeit, Annahmen zu revidieren und Wissen dynamisch anzupassen, unterstützt nicht-monotones Schließen die Entwicklung komplexer, adaptiver Systeme, die robust und reaktionsfähig sind, vor allem in Bereichen mit hohem Maß an Unsicherheit und Veränderung. Diese Problemlösungsstrategie wird häufig auch als Truth Maintenance System (TMS) bezeichnet und ist im medizinisch, diagnostischen Bereich weit verbreitet.

Temporales Schließen ist eine Problemlösungsstrategie, die sich mit dem Verständnis und der Modellierung zeitabhängiger Informationen befasst. Sie bezieht sich auf die Fähigkeit, Schlussfolgerungen über zeitliche Beziehungen zwischen Ereignissen oder Zuständen zu ziehen. Diese Art des Schließens ist besonders relevant in Systemen, die mit dynamischen oder zeitveränderlichen Daten arbeiten, wie zum Beispiel bei der Planung, Überwachung und Vorhersage von Ereignissen.

Temporales Schließen basiert oft auf zeitlichen Logiken, wie der Temporalen Logik oder der linearen, temporalen Logik, wie Allen sie bereits 1983 etabliert hat[75]. Diese Logiken ermöglichen die Formulierung und Überprüfung von Aussagen über zeitliche Abläufe, wie „Ereignis A passiert vor Ereignis B" oder „Zustand C wird immer während des Intervalls D wahr sein".

Zur Unterstützung des temporalen Schließens werden häufig spezielle, zeitliche Objektstrukturen verwendet, wie Zeitstempel oder Zeitintervalle, um Ereignisse oder Zustände im zeitlichen Kontext zu repräsentieren.

Die Intervallarithmetik ist eine Methodik, um mit Unsicherheiten in zeitlichen Daten umzugehen, indem Ereignisse oder Zustände nicht als feste Zeitpunkte, sondern als Intervalle modelliert werden, die eine gewisse Flexibilität in ihrer zeitlichen Platzierung erlauben.

Das temporale Schließen erweitert die klassischen Logiksysteme um die Dimension der Zeit, was es ermöglicht, komplexe dynamische Systeme zu modellieren und zu analysieren. Es stellt sicher, dass Systeme nicht nur aufgrund statischer Information, sondern auch basierend auf der zeitlichen Entwicklung von Daten und Ereignissen Entscheidungen treffen können.

Objektorientierte Darstellungen beziehen sich auf den Einsatz von Prinzipien und Konzepten der Objektorientierung zur Modellierung und Lösung von Problemen im Bereich der wissensverarbeitenden Systeme. Sie verwenden Grundkonzepte der Objektorientierung wie Klassen, Objekte, Vererbung, Kapselung und Polymorphismus.

In der Wissensrepräsentation werden diese Grundkonzepte verwendet, um komplexe Wissensstrukturen zu modellieren, wie beispielsweise Ontologien in semantischen Netzen oder Agenten in Multi-Agenten-Systemen. Der Entwurf und die Implementierung objektorientierter Systeme können allerdings komplex und zeitaufwändig sein, im Besonderen bei großen und vernetzten Systemen. Vorteile wie Modularität, Wartbarkeit, Wiederverwendbarkeit und Flexibilität wiegen dies jedoch auf.

Für die Modellierung und Ausführung klinisch-therapeutischer Prozesse wird eine einzige Problemlösungsstrategie allein nicht ausreichen. Vielmehr erfordern die

dort charakteristischen Wissensinhalte und deren Verknüpfungen eine alternierende Kombination mehrerer Problemlösungsstrategien. Für die Implementierung der in Tabelle 2.3, Seite 23 angeführten Beispiele klinisch-therapeutischer Prozesse haben wir die Kombination der Problemlösungsstrategien regelbasiertes Schließen, probabilistisches Schließen, nicht-monotones Schließen und temporales Schließen verwendet.

3.5 KNOWLEDGE WIKIS

Ein Knowledge Wiki ist eine kollaborative, digitale Plattform, die auf der Wiki-Technologie basiert. Diese ermöglicht Benutzern, Wissen in Form von Hypertext-Dokumenten online als sogenannte Artikel zu erstellen, zu bearbeiten und zu organisieren. Artikel sind in der Regel in einer einfachen Markup-Sprache verfasst und erlauben es, Informationen auf flexible Weise zu verknüpfen und zu strukturieren.

Die zugrunde liegende Technologie eines Knowledge Wikis basiert auf einem Web-Content-Management-System, das es theoretisch unbegrenzt vielen Benutzern ermöglicht, Inhalte simultan zu bearbeiten. Ein bekanntes Beispiel ist die Software *MediaWiki*, die u. a. von Wikipedia verwendet wird. Multimedia-Objekte jeglicher Art und Form können ebenfalls einfach in einem Knowledge Wiki platziert und verknüpft werden.

Eine der Hauptstärken eines Knowledge Wikis ist die Möglichkeit der kollaborativen Bearbeitung. Benutzer können Inhalte hinzufügen, aktualisieren oder ändern, was die kontinuierliche Weiterentwicklung und Aktualisierung des gesammelten Wissens fördert. Änderungen im Wiki werden protokolliert, und frühere Versionen von Artikeln bleiben zugänglich. Dies ermöglicht es, den Verlauf von Änderungen nachzuverfolgen und gegebenenfalls frühere Versionen wiederherzustellen. Artikel in einem Knowledge Wiki können in hierarchischen Kategorien organisiert werden, um die Navigation und das Auffinden von Informationen zu erleichtern. Die Fähigkeit, Artikel über Hyperlinks miteinander zu verbinden, ermöglicht die Erstellung eines Netzwerks von Wissen, in dem Benutzer leicht zwischen verwandten Themen navigieren können.

Viele Organisationen und Unternehmen nutzen Knowledge Wikis, um internes Wissen zu dokumentieren und zugänglich zu machen, was die Einarbeitung neuer Mitarbeiter erleichtert und die Zusammenarbeit verbessert. Forscher verwenden Wikis, um Projekte zu dokumentieren, Forschungsdaten zu sammeln, auszuwerten und zu publizieren sowie interdisziplinäre Zusammenarbeit zu fördern.

Knowledge Wikis bieten eine flexible und skalierbare Plattform zur Wissensverwaltung und -verbreitung, die die kollektive Intelligenz einer Gruppe oder Gemeinschaft nutzt. Sie fördern zudem eine offene, agile Zusammenarbeit und ein kontinuierliches Lernen, was zu einer dynamischen und ständig wachsenden Wissensbasis führt. Durch die intuitive Bedienbarkeit von Wikis wird die Hemmschwelle für die Partizipation gesenkt, was die Beteiligung einer breiten Benutzerbasis ermöglicht und die Vielfalt der gesammelten Informationen erhöht. In der Wissenschaft unterstützt dies den offenen Austausch von Wissen und Daten, was die Forschung und Innovation vorantreibt.

Mit ihrer zentralisierenden Bündelung von Wissen ganz unterschiedlicher Art und Form bieten sich Knowledge Wikis geradezu an, das Prinzip einer singulären Datenquelle umzusetzen, was für diese Arbeit von exponierter Bedeutung ist. Knowledge Wikis liefern also eine anpassbare, hybride Technologie, indem sie Wissensrepräsentationsformate, Problemlösungsstrategien, semantikorientierte Objektdarstellungen für Ontologien und versionskontrolliertes Dokumentenmanagement vereinen.

Werden Knowledge Wikis dann noch erweitert um Funktionalitäten zur Programmierung innerhalb der modellierten Wissensinhalte, so entstehen hoch-funktionale und mächtige Entwicklungsumgebungen für wissensverarbeitende Systeme. Ein solches Knowledge Wiki, dass zudem auch moderne Methoden der Software-Entwicklung wie agiles Vorgehen, kontinuierliche Integration, etc. unterstützt, stellen Puppe et al. mit KnowWE vor [77].

Da Knowledge Wikis systemisch Wissen von Algorithmen trennen und eine Vielzahl von Repräsentationsformaten zur Darstellung von Wissen offerieren, sind wissensbasierte Systeme grundsätzlich geeignet, dem hoch wissensintensiven, komplexen Charakter der Domäne Anästhesiologie gerecht zu werden. Weiterer Komfort ergibt sich aus den unterschiedlichen Problemlösungsstrategien, die die Komplexität klinisch-therapeutischer Prozesse beherrschbar machen. Besonders wenn sie miteinander kombiniert werden können.

Die sehr früh auf dem Lösungsweg erfolgte Ergänzung des basalen Strukturparadigmas für Expertensysteme um multiple Wissensbasen, siehe Abbildung 3.1(c), ist neuartig und ermöglicht die spätere Nutzung des Software-Frameworks für Entwicklung und Betrieb einer Vielzahl von wissensbasierten Software-Anwendungen.

Knowledge Wikis erweitern klassische Expertensysteme um die Möglichkeit der interaktiven, verteilten Zusammenarbeit von Projektbeteiligten und ermöglichen so eine frühe und kontinuierliche Kollaboration mit Domänenexperten, also den Ärzten und dem Pflegepersonal.

KAPITEL 4

METHODIK

Nun, da wir sowohl die Problemstellung, den Anwendungsbereich der Anästhesiologie, die klinisch-therapeutischen Prozesse als auch die allgemeine Methodik der wissensverarbeitenden Systeme kennengelernt haben, wenden wir uns dem zentralen Thema der zur Problemlösung erarbeiteten, konkreten Methodik zu.

Das Ziel dieses Kapitels ist es, ein grundlegendes Verständnis darüber zu schaffen, wie wir die Problemstellung methodisch lösen wollen, ohne auf etwaige technologische Aspekte einzugehen.

Die Festlegung aller Systemanforderungen über das Anforderungsmanagement führen zum Entwurf einer adäquaten Systemarchitektur inklusive der Prinzipien zur zeitlichen Dynamik und Steuerung des Gesamtsystems. Insbesondere die flexible Systemarchitektur leitet die spätere Implementierung und wird die angestrebten Qualitätsattribute zukunftsfähig realisieren. Daher wird das ihr zu Grunde liegende Modell und dessen funktionale Eigenschaften eingehend beleuchtet.

Aus den zentralen Systemanforderungen und den bereits vorgestelltem Leit- und Design-Prinzip haben wir sukzessive drei Paradigmen erarbeitet, die einen qualitätssichernden Rahmen spannen und der Orientierung aller weiteren Entwicklungstätigkeiten dienen. Sie werden detailliert beschrieben und in Beziehung gesetzt, um im weiteren Verlauf der Lösungserarbeitung immer wieder herangezogen und überprüft werden zu können.

Auf diese Art und Weise sind wird in der Lage, in jeder Phase der Entstehung des Software-Frameworks die bis dahin erreichten Ergebnisse zu validieren und gegebenenfalls zu korrigieren.

4.1 EINORDNUNG IN FORSCHUNG UND TECHNIK

In Abschnitt 1.3 haben wir bereits methodische Ansätze vorgestellt, die sich mit der Computerisierung von medizinischen Leitlinien beschäftigen und dabei diese beiden Zielstellungen verfolgen:

- Digitalisierung von klinisch-therapeutischen Prozessen via medizinischer Leitlinien
- Bereitstellung interaktiver Entwicklungsumgebungen für medizinische Leitlinien, sogenannte Autorensysteme.

Die basalen Prinzipien dieser Methodiken können wir konzeptuell aufgreifen, jedoch sind sie zur vollständigen Erfüllung der zentralen Anforderungen dieser Arbeit, siehe Abschnitt 1.4, unzureichend.

So vollzieht sich die Digitalisierung medizinischer Leitlinien zumeist über die Nutzung textueller Programmiersprachen (engl. Textual Programming Language (TPL)) oder einfacher graphischer Notationen wie zum Beispiel des Guideline Interchange Format (GLIF).

Mit dem Leitprinzip *Vom Programmieren zum Spezifizieren* streben wir den Idealfall an, dass der Domänenexperte nahezu die gesamte Spezfikationsarbeit leisten kann. Dazu ist es zwingend notwendig, ihm die Bereitstellung seines prozessualen Wissens so einfach und intuitiv wie möglich zu gestalten. In den vergangenen Jahren wurde mehrfach die Verwendung der Unified Modelling Language (UML) untersucht und empfohlen. Insbesondere die umfangreichen Projektaktivitäten der TU Ilmenau, die zeitlich in etwa mit dem Entstehen des Software-Frameworks zusammenfallen, sind hier zu nennen [79]. Jedoch kann die Modellierungssprache UML mit ihrer Komplexität und eher technischen Ausrichtung dem o. g. Anspruch der Einfachheit im klinischen Umfeld nicht gerecht werden. Das gilt ebenso für moderne Ansätze sogenannter visueller Programmiersprachen (engl. Visual Programming Language (VPL)), derer es inzwischen viele gibt, die allerdings ebenso komplex und technisch sind. Siehe hierzu auch das umfassende Review von Idrees und Aslam aus dem Jahre 2022 [84].

Im Sinne einer VPL haben wir daher eine eigene graphische Notation eingeführt, die sich in eine vorhandene Entwicklungsumgebung integrieren lässt und sämtliche Anforderungen an die Gebrauchstauglichkeit et al. erfüllt und damit zu einer eigenen VPL wird.

Neuartig ist auch, dass sich die Ausführung digitalisierter medizinischer Leitlinien nicht nur auf den Kontext der jeweiligen Entwicklungsumgebung beschränkt, sondern vielmehr in beliebigen Medizinprodukten möglich ist.

Resümierend könnten wir damit das Software-Framework als eine auf klinisch-therapeutische Prozesse für Medizinprodukte spezialisierte Form einer Low-Code/No-Code Plattform (LCNC) bezeichnen, so wie sie beispielsweise jüngst in Tripathi definiert wird [54].

Die Umsetzung der zentralen Forderung nach einer Multiplzität über zwei Freiheitsgrade, siehe ebenfalls Abschnitt 1.4, ist inspiriert durch das Universal Computer-oriented Language (UNCOL)-Konzept. UNCOL wurde bereits 1958

von Conway eingeführt [85] und schlägt die Lösung des $M * N$ Problems über die Nutzung einer Zwischensprache vor, wodurch die Komplexität der Aufgabe auf ein $M + N$ reduziert wird. Prominent genutzt wurde und wird dieses Konzept im Compiler-Bau. In unserem Fall entspricht der Zwischensprache das Software-Framework selbst.

Die Ausführung zuvor computerisierter klinisch-therapeutischer Prozesse in einer Vielzahl von Medizinprodukten bedarf einer flexiblen, komponenten-basierten, wohlstrukturierten Systemarchitektur, die das Paradigma *Trenne invariante von variablen Anteilen* realisiert. Recherchen zum Zeitpunkt der Entstehung des Software-Frameworks wie auch zum gegenwärtigen Zeitpunkt, konnten keinerlei vergleichbare Ansätze hervorbringen. So dürfte die hier vorgestellte Systemarchitektur mit ihrer einfachen Erweiterbarkeit und Anbindung theoretisch beliebiger Zielsysteme ein Novum sein.

4.2 ANFORDERUNGSMANAGEMENT

Die zuvor betrachtete Methodik und Funktionsweise wissensverarbeitender Systeme hat die Entscheidung für diese Technologie zur Entwicklung des Software-Frameworks (SWFW) argumentiert. Es entsteht eine Infrastruktur zur Entwicklung wissensbasierter Software-Anwendungen (WBSA) für den Einsatz in der Anästhesiologie.

Zu unterscheiden ist generell zwischen Produktanforderungen des Software-Frameworks und denen der damit entwickeltem wissensbasierten Software-Anwendungen. An dieser Stelle beschreiben wir detailliert und ausschließlich die funktionalen Anforderungen an das Software-Framework für die Ausführung wissensbasierter Software-Anwendungen. Die Produktanforderungen der mit dem Software-Framework entwickelten Software-Anwendungen ergeben sich aus der Spezifikation und der Modellierung des zugrundeliegenden klinisch-therapeutischen Prozesses (KTP) im Knowledge Wiki.

Üblicherweise beginnt die Entwicklung eines Medizinproduktes mit der Erstellung einer Zweckbestimmung (engl. Intended Use), gefolgt von einem Lastenheft, das u. a. die funktionalen und nicht-funktionalen Systemanforderungen festlegt.

Für das Software-Framework definieren wir die Zweckbestimmung wie folgt:

Definition ZWECKBESTIMMUNG SWFW

> Entwurf, Design, Konstruktion, Transfer, Ausführung sowie Analyse, Wartung und Pflege von wissensbasierten Software-Anwendungen in der Anästhesiologie, die viele klinisch-therapeutische Prozesse für viele Zielsysteme flexibel, effizient und weiterverwendbar realisieren

Mit dem Anforderungsmanagement (engl. Requirements Engineering) werden sämtliche Funktionalitäten, Eigenschaften, Einsatzbereiche, Restriktionen, Rahmenbedingungen u. v. m. präzise und verbindlich zwischen Auftraggeber und Auftragnehmer definiert und im Lastenheft dokumentiert freigegeben. Auftraggeber für die Entwicklung des Software-Frameworks war eine unternehmensinterne

Organisationseinheit, das Produktmanagement, Auftragnehmer die ebenfalls interne Entwicklungsabteilung. Wichtig und späteren Konflikten vorbeugend ist eine möglichst umfassende und konkrete Produktspezifikation, inklusive der protokollierten Freigabe aller Beteiligten, bevor die eigentliche Produktentwicklung beginnt. Leider läuft dies jedoch der Tatsache zuwider, dass zu frühen Projektzeiten häufig viele Anforderungen noch unklar und vage sind. Inkrementell, evolutionäre Ansätze sowie agile Methoden der Software-Entwicklung können hier Abhilfe schaffen.

Die grafische Modellierung als Anwendungsfälle (engl. Use Cases) hat sich bewährt, um die funktionalen Anforderungen auf abstrakter Ebene zu spezifizieren. Anwendungsfälle sind eine Technik zur Erfassung der funktionalen Systemanforderungen. Das dabei entstehende Anwendungsfalldiagramm (engl. Use Case Diagram) kann mit dem technisch bisweilen wenig affinen Auftraggeber leicht verständlich diskutiert werden und dient gleichsam als Einstiegspunkt für die weitere Verfeinerung und Detaillierung in einem Software-Modell. Es ist eine externe Sicht auf das zu entwickelnde Produkt.

Abbildung 4.1 zeigt alle Anwendungsfälle des Software-Frameworks in der dafür etablierten Unified Modelling Language (UML) [39]. In der UML wird ein Akteur durch ein Strichmännchensymbol dargestellt, unabhängig davon, ob der Akteur ein Mensch, ein externes System oder ein Gerät ist.

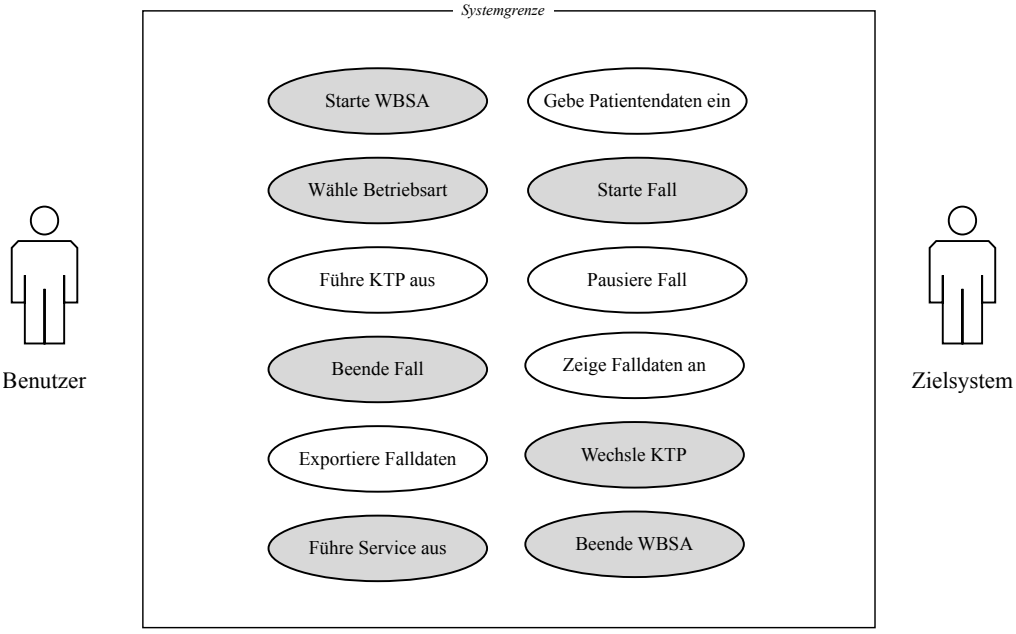

ABBILDUNG 4.1 – *Anwendungsfalldiagramm des Software-Frameworks*

Ein Design-Prinzip des Software-Frameworks ist die konsequente Trennung von invarianten und variablen Anteilen. Dies beginnt schon bei den Anwendungsfällen. Invariante und damit unverändert wieder- und weiterverwendbare Anwendungsfälle sind in Abbildung 4.1 grau hinterlegt. Alle anderen sind variabel, also spezifisch für jeden klinisch-therapeutischen Prozess.

Die als Akteure bezeichneten *Benutzer* und *Zielsystem* stehen mit allen Anwendungsfällen in bidirektionaler Interaktion, daher wurde auf die entsprechenden Verbindungslinien aus Gründen der Lesbarkeit verzichtet. Die wissensbasierte Software-Anwendung selbst ist im Zielsystem verortet und wird dort ausgeführt, was der typischen Konstellation im klinischen Alltag entspricht.

Die Anwendungsfälle sind in einer kausalen und chronologischen Reihenfolge angeordnet, die von links nach rechts und oben nach unten zu lesen und wie folgt zu interpretieren ist. Dabei bedeutet ein Patientenfall - oder kurz: Fall - die Ausführung des klinisch-therapeutischen Prozesses für einen spezifischen Patienten.

Die wissensbasierte Software-Anwendung wird vom *Benutzer* gestartet und von diesem sowohl mit Patientengrunddaten als auch der gewünschten Betriebsart beschickt. Anschließend kann der Benutzer den für einen spezifischen Patienten vorbereiteten Fall starten, während dem zyklisch der klinisch-therapeutische Prozess, i. e. die Wissensbasis, abgearbeitet wird, bis der Fall schließlich zum Ende kommt. Alle weiteren Anwendungsfälle finden außerhalb eines Patientenfalles statt. Im Einzelnen:

Starte WBSA. Die im jeweiligen Zielsystem, z. B. Beatmungsgerät, Anästhesie-Arbeitsplatz, Inkubator, Patientendaten-Managementsystem eingebettete WBSA soll jederzeit vom Benutzer, i. e. Arzt, Pflegepersonal, über das Graphical User Interface (GUI) gestartet werden können, sobald das Zielsystem betriebsbereit ist.

Gebe Patientendaten ein. Die vom jeweiligen KTP vor Fallbeginn erwarteten Patientengrunddaten sollen via GUI an das WBSA gegeben werden. Patientengrunddaten sind abhängig vom klinisch-therapeutischen Prozess und spezifizieren für gewöhnlich Informationen zu Anamnese, therapeutischen Rahmenbedingungen und/oder eventuellen Kontraindikationen.

Wähle Betriebsart. Dem Benutzer soll die Möglichkeit geboten werden, zwischen vordefinierten Betriebsarten der WBSA zu wählen, die die Invasivität der schlussfolgernden, therapeutischen Aktionen des klinischen Prozesses festlegen. Die drei möglichen Betriebsarten sind: Entscheidungsunterstützung, Halbautomatik oder Vollautomatik, siehe auch Abschnitt 1.4.

Starte Fall. Der gewünschte klinisch-therapeutische Prozess ist gewählt, ebenso die gewünschte Betriebsart, die Patientengrunddaten sind ebenfalls eingeben worden. Nun soll der Fall, also die Ausführung des klinisch-therapeutischen Prozesses (i. e. Wissensbasis) beginnen. Auch dieser Anwendungsfall wird vom Benutzer explizit ausgelöst.

Führe KTP aus. Gemäß der in der jeweiligen Wissensbasis modellierten Spezifika soll der ausgewählte klinisch-therapeutische Prozess im Zielsystem ausgeführt werden, wobei der Befund-Diagnose-Ziel-Therapie Zyklus periodisch mit einer a priori vordefinierten Taktrate instantiiert und durchlaufen werden soll.

Pausiere Fall. Beim Eintreffen eines oder mehrerer vordefinierter Ereignisse soll ein aktiver Fall ausgesetzt werden (Leerlauf), indem die Ausführung des klinisch-therapeutischen Prozesses angehalten wird. Wenn alle Ereignisse verflüchtigt

sind, soll der Fall selbsttätig fortgesetzt werden. Solche Ereignisse können Benutzeranforderungen, technische (lösbare) Probleme oder anwendungsspezifische, unkritische Situationen sein.

Zeige Falldaten an. Während der Ausführung des klinisch-therapeutischen Prozesses soll der Benutzer jederzeit Gelegenheit haben, sich proaktiv über den aktuellen Stand des Falles und dessen Historie zu informieren. Dazu gehören mindestens: Eingehende Messwerte, Einstellwerte, Alarm- und Zustandsmeldungen des Zielsystems, ausgehende Einstellwerte, Alarm- und Zustandsmeldungen.

Beende Fall. Der Benutzer soll via GUI des Zielsystems jederzeit einen laufenden Patientenfall beenden können. Zudem sollen kritische, technische und/oder KTP-spezifische Probleme einen Fall intrinsisch beenden.

Wechsle KTP. Der Benutzer soll via GUI des Zielsystems jederzeit einen anderen klinsch-therapeutischen Prozess zur Ausführung selektieren können. Dies soll jedoch nur möglich sein, wenn kein Patientenfall aktiv ist. Technisch gesehen ist dies der Wechsel zwischen verschiedenen Wissensbasen.

Exportiere Falldaten. Alle zu einem Patientenfall aufgezeichneten Falldaten sollen in einem persistenten Format aus dem Zielsystem exportiert werden.

Beende WBSA. Der Benutzer soll via GUI des Zielsystems jederzeit eine laufende wissensbasierte Software-Anwendung beenden können.

Führe Service aus. Für die Wartung einer wissensbasierten Software-Anwendung wichtige Informationen sollen abrufbar sein. Dazu gehören in der Regel Systeminformationen und exportierbare Fall- und Fehlerdaten. Außerdem soll es möglich sein, eine wissensbasierte Software-Anwendung zu konfigurieren und dessen Firmware (System-Software) zu aktualisieren. Die Service-Funktionen sollen jedoch nur nutzbar sein, wenn keine wissensbasierte Software-Anwendung aktiv ist.

Neben den funktionalen Anforderungen ist es wichtig für den späteren Entwurf einer adäquaten Systemarchitektur, die nicht-funktionale Eigenschaften festzulegen. Sowohl unmittelbar für das Software-Framework als auch mittelbar für jede damit entwickelte wissensbasierte Software-Anwendung sollen gemäß ISO/IEC 25010 [111] die folgenden Software-Qualitätsattribute erfüllt werden:

Gebrauchstauglichkeit (engl. Usability) beschreibt die Benutzerfreundlichkeit und Effizienz, mit der Benutzer eine Software-Anwendung erlernen, verstehen und nutzen können, um ihre spezifischen Ziele zu erreichen. Eine hohe Gebrauchstauglichkeit ist entscheidend, um sicherzustellen, dass Benutzer mit der Software zufrieden sind und ihre Aufgaben effektiv und effizient erledigen können.

Portierbarkeit beschreibt die Fähigkeit einer Software, mit minimalem Aufwand von einer Umgebung in eine andere übertragen zu werden. Diese Umgebungen können sich in verschiedenen Aspekten unterscheiden, wie etwa Hardware-Plattformen, Betriebssystemen oder Software-Abhängigkeiten. Portierbarkeit ist ein wichtiges Kriterium, um die langfristige Wartbarkeit und Flexibilität von Software-Anwendungen zu gewährleisten.

Erweiterbarkeit beschreibt die Fähigkeit eines Software-Systems, mit minimalem Aufwand um neue Funktionen oder Merkmale erweitert zu werden. Erweiterbarkeit ist ein entscheidender Faktor, um sicherzustellen, dass Software-Anwendungen

langfristig relevant und anpassungsfähig bleiben gegenüber sich ändernden Anforderungen oder Technologien.

Interoperabilität beschreibt die Fähigkeit eines Software-Systems, effektiv mit anderen Systemen, Anwendungen oder Komponenten zu kommunizieren und zu kooperieren. Interoperabilität ist von ausschlaggebender Bedeutung, um sicherzustellen, dass unterschiedliche Systeme nahtlos zusammenarbeiten und Daten oder Dienste austauschen können.

Wartbarkeit beschreibt die Fähigkeit eines Software-Systems, Änderungen effizient und effektiv vorzunehmen. Diese Änderungen können Fehlerbehebungen, Updates, Verbesserungen oder Anpassungen an neue Anforderungen umfassen. Wartbarkeit ist bedeutsam für den langfristigen Erfolg und die Nachhaltigkeit einer Software-Anwendung, da sie direkt die Kosten und den Aufwand für die Pflege und Weiterentwicklung der Software beeinflusst.

Robustheit beschreibt die Fähigkeit eines Software-Systems, unter ungewöhnlichen oder fehlerhaften Bedingungen dennoch zuverlässig zu funktionieren. Robustheit ist essenziell für die Gewährleistung der Stabilität und Zuverlässigkeit einer Software, auch wenn sie mit unvorhergesehenen Situationen konfrontiert wird, wie z. B. ungültigen Eingaben, Systemfehlern oder unerwarteten Benutzeraktionen.

Die funktionalen und nicht-funktionalen Produktanforderungen sind entscheidende Faktoren für die Auslegung einer adäquaten Systemarchitektur, deren Design jedoch ebenso vom Leitprinzip *vom Programmieren zum Spezifizieren* geführt sein muss.

4.3 SYSTEMARCHITEKTUR

Um eben diesem Leitprinzip gerecht werden zu können, bei gleichzeitiger Zusicherung von Stabilität und Flexibilität, haben wir uns dem Instrument der Entwurfsmuster (engl. Design Pattern) bedient, wie sie beispielsweise von Bass et al. [35] oder dem Standardwerk von Gamma [27] beschrieben werden.

Folgende drei Entwurfsmuster kamen zur Anwendung:

Komponentenunabhängigkeit. Alle Komponenten des Systems werden unabhängig voneinander und einzeln betrachtet und behandelt. Sie wissen theoretisch nichts voneinander und werden als vertikale Einschübe, sogenannte *Slots*, modelliert. Solch ein Slot-Modell ist mit minimalem Aufwand beliebig erweiterbar und verbessert vor allem Erweiterbarkeit und Skalierbarkeit.

Mehrschichtmodell. Auch *N-Tier* genannt. Horizontal zu den Slots werden nun Schichten (Tiers) eingezogen, womit sich die Möglichkeit eröffnet, trotz der gewünschten Komponentenunabhängigkeit gemeinsame Funktionalitäten über alle Slots zu führen. Typische Beispiele für Schichten sind Vermittlungsschicht, Anwendungsschicht, Datenschicht und Präsentationsschicht, wie sie z. B. in Fowler aufgeführt sind [38]. Mehrschichtmodelle erhöhen signifikant Flexibilität, Modularität, Skalierbarkeit sowie Wiederverwendbarkeit eines Systems.

Virtuelle Maschine. Eine prozessbasierte, virtuelle Maschine stellt eine vom Rest der Systemarchitektur unabhängige Umgebung auf einem Zielsystem zur Verfü-

gung, in der eine spezifische Funktionalität isoliert, aber dennoch in Interaktion mit dem Rest des Systems, ausführbar ist. Dieses Entwurfsmuster bietet den großen Benefit, dass mit ihm die Portierung auf eine Vielzahl von Zielsystemen mit unterschiedlichen Betriebssystemen einfach machbar wird. Außerdem verbessern prozessbasierte, virtuelle Maschinen die Sicherheit eines Systems erheblich, allerdings zum Preis eines erhöhten Betriebsmittelverbrauchs.

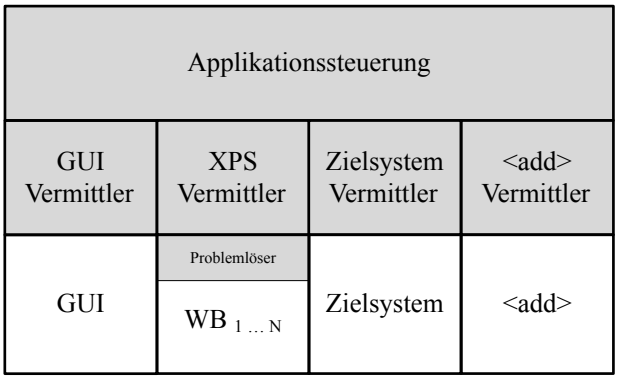

ABBILDUNG 4.2 – *Basis-Systemarchitektur des Software-Frameworks*

Damit entstand die in Abbildung 4.2 zu sehende Basisarchitektur, die wir nachfolgend beschreiben.

Die Systemarchitektur besteht aus drei Schichten, von denen die *Applikationssteuerung* als oberste die gesamte Steuerung während der Laufzeit übernimmt. Die nächste Schicht ist eine *Vermittlungsebene*, mit der die bidirektionale Kommunikation vertikaler Slots erfolgen kann, ohne dass diese direkt miteinander kommunizieren müssen. Können Slots systemisch bedingt nicht horizontal miteinander agieren - sie wissen quasi nichts voneinander, ist es einfach, Slots zu ersetzen und/oder neue hinzuzufügen. So ist auch in der Dynamik des Systems die Unabhängigkeit der Komponenten garantiert. Die dritte Schicht schließlich ist die *Anwendungsschicht*, in der jeder Slot mit seiner Funktionalität zur Umsetzung der Anwendungsfälle beiträgt.

Es sind dies: Das *GUI* Graphical User Interface der wissensbasierten Software-Anwendung, nicht zu verwechseln mit dem GUI des jeweiligen Zielsystems, auch wenn diese häufig miteinander verwoben sind. Mit dem Slot *Zielsystem* erfolgt die datentechnische Anbindung zwischen wissensbasierter Software-Anwendung und der involvierten Medizintechnik, also beispielsweise einem Beatmungsgerät. Der Slot *XPS* (Expertensystem) deckt den gesamten Bereich der wissensverarbeitenden Systemkomponente ab. Zu beachten ist, dass der XPS-Slot fähig ist, mit mehreren Wissensbasen (*WB*) umzugehen, obwohl er mit nur einem *Problemlöser* auskommen muss. Der Problemlöser ist gleichbedeutend mit der Problemlösungskomponente aus Abbildung 3.2, Seite 31. Die prozessbasierte, virtuelle Maschine ist ausschließlich und alleinig der Anwendungsschicht des XPS-Slot vorbehalten, denn genau hier erfolgt die Demarkierung zwischen wissensverabeitendem und konventionellem System.

Wir bezeichnen die einzelnen Funktionsblöcke der Basis-Systemarchitektur als *Subsysteme*, exklusive der Problemlösungsstrategie (Abschnitt 3.4), die in der Architektur kurz *Problemlöser* genannt wird und zum Subsystem XPS gehört.

Die flexible Erweiterbarkeit des Systems um zusätzliche Slots, die ihrerseits wieder aus Vermittlungs- und Anwendungsschicht bestehen müssen, ist durch den Einschub $< add >$ angedeutet. Typische Beispiele für weitere Slots sind:

- Anbindung weiterer Medizinprodukte, die nicht als Zielsysteme fungieren
- Interaktion mit dem Krankenhausinformationssystem
- Anbindung weiterer medizinischer Software, wie beispielsweise ein PDMS
- Anbindung weiterer GUI
- Testfunktionalitäten
- Unterstützung wissenschaftlicher Forschungsaktivitäten

Die Anbindung an ein Krankenhausinformationssystem eröffnet u. a. die Möglichkeit, nicht nur klinisch-therapeutische, sondern auch klinisch-administrative Prozesse zu computerisieren.

Invariante Komponenten der Basis-Systemarchitektur sind in Abbildung 4.2 grau hinterlegt. Ist das Software-Framework erst einmal erschaffen, so gilt für jede damit entwickelte wissensbasierte Software-Anwendung, dass für alle zukünftigen Erweiterungen, Änderungen, Korrekturen etc. lediglich die in weiss dargestellten Anteile betroffen sind. Ersichtlich wird der enorme Vorteil, den ein hoher Grad an invarianten Systemkomponenten mit sich bringt.

Applikationssteuerung. Wie erwähnt, übernimmt die oberste Schicht der Systemarchitektur die permanente Kontrolle des Laufzeitverhaltens während der gesamten Ausführung. Sie ist vergleichbar mit einer Vermittlungszentrale. Anfragen, Daten und Ereignisse gehen von einzelnen Slots ein und müssen von der Anwendungssteuerung weitergereicht und beantwortet werden. Als Steuerungsmechanismen für die Aufgaben der Vermittlungszentrale kommen zum Einsatz:

- Zeitsteuerung - alle N Zeiteinheiten wird eine Inferenz im Slot XPS ausgelöst
- Ereignissteuerung - der Benutzer bedient zu einem beliebigen Zeitpunkt das GUI
- Synchronität - für die Dauer einer Service-Funktion müssen alle internen Prozesse zeitlich koordiniert werden
- Asynchronität - eine unerwartete Fehlersituation tritt auf und muss sukzessive abgefangen werden

Zur Ereignissteuerung ist anzumerken, dass zusammen mit dem Versenden eines Ereignisses (engl. Event) auch Nutzdaten hinzugefügt werden können.
Beispiel: Der Benutzer hat via GUI Falldaten angefordert (Anwendungsfall *Zeige Falldaten an*). Dann wird zusammen mit dem Ereignis auch weitergegeben, welcher Fall gewünscht ist.

Fundamentale, inhärente Systemeigenschaften des Software-Frameworks werden in den folgenden Abschnitten in Form von Paradigmen festgelegt.

4.4 PARADIGMA I - ZWEI FREIHEITSGRADE

Bei der Beschäftigung mit dem ursprünglichen Entwicklungsauftrag der Firma Dräger, eine bestehende, akademische Software-Anwendung zur teil-automatisierten Entwöhnung von der Beatmung [83] zu portieren und zu kommerzialisieren, entstand die Frage „Wie kann ein Software-System mehr leisten, als eine reine 1:1 Umsetzung?". Der Wunsch war, möglichst einfach und effizient und ohne großen Programmieraufwand, weitere Software-Anwendungen dieser Art zu entwickeln bzw. bereits bestehende zu ändern. Das führte zu einem Gesamtkonzept, dessen Paradigmen genau diese Frage beantworten.

In Abschnitt 1.4 haben wir das Prinzip der Multiplizität kennengelernt, das eine zentrale Eigenschaft des Software-Frameworks liefern soll, mit der dann auch ökonomisch ein Mehrwert erzielbar ist. Multiplizität meint in diesem Fall die Fähigkeit des Software-Frameworks, wissensbasierte Software-Anwendungen für viele klinisch-therapeutische Prozesse (Freiheitsgrad 1) zu entwickeln, die in beliebigen Zielsystemen ablauffähig sind (Freiheitsgrad 2). Der Terminus „beliebig" ist hier eher theoretisch gemeint, denn sowohl klinisch-therapeutische Prozesse als auch die jeweiligen Zielsysteme unterliegen gewissen Rahmenbedingungen, die erfüllt sein müssen, um zu einer wissensbasierten Software-Anwendung zu werden. Es entstehen somit eine generative Kombinatorik sowie zwei Freiheitsgrade der Flexibilität, die in Abbildung 4.3 illustriert sind.

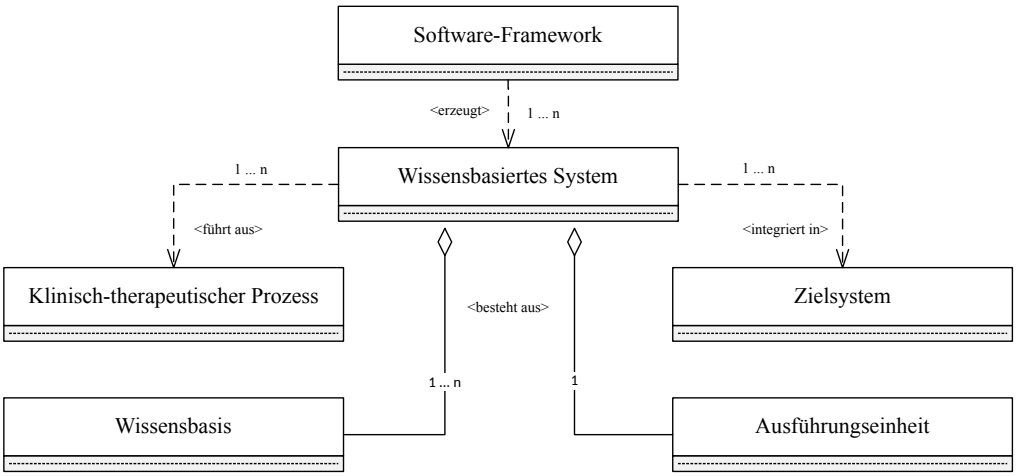

ABBILDUNG 4.3 – *Wirkprinzip des Software-Frameworks mit Basiskomponenten*

Mit dem Software-Framework können ein oder mehrere wissensbasierte Software-Anwendungen für ein oder mehrere Zielsysteme entwickelt werden. Die jeweilige wissensbasierte Software-Anwendung wiederum kann ein oder mehrere zuvor computerisierte (i. e. modellierte) klinisch-therapeutische Prozesse in ein oder mehreren Zielsystemen ausführen. Es ergeben sich also folgende Konstellationen für eine wissensbasierte Software-Anwendung:

1. ein klinisch-therapeutischer Prozess wird in genau einem Zielsystem aus-

geführt
2. ein klinisch-therapeutischer Prozess wird in mehreren Zielsystemen ausgeführt
3. mehrere klinisch-therapeutische Prozesse werden in genau einem Zielsystem ausgeführt
4. mehrere klinisch-therapeutische Prozesse werden in mehreren Zielsystemen ausgeführt

Technische Voraussetzung für die Multiplizität ist zum einen die Fähigkeit einer wissensbasierten Software-Anwendung, mehrere Wissensbasen (i. e. klinisch-therapeutische Prozesse) disjunkt aufnehmen zu können und zum anderen, dies mit genau einer sogenannten *Ausführungseinheit* (engl. Runtime Engine) zu leisten. Der invariante Anwendungsfall *Wechsle KTP* realisiert die Umschaltung zwischen den Wissensbasen, der Anwendungsfall *Führe KTP aus* die Ausführungseinheit. Diese beiden Basiskomponenten einer wissensbasierten Software-Anwendung repräsentieren die *Problemlösungskomponente* sowie das *bereichsbezogene Expertenwissen* aus der allgemeinen Architektur eines Expertensystems (Abbildung 3.2, Seite 31) und sind ebenfalls in Abbildung 4.3 dargestellt.

Das Paradigma der zwei Freiheitsgrade realisiert im Besonderen die Software-Qualitätsattribute Portierbarkeit, Erweiterbarkeit und Interoperabilität.

4.5 PARADIGMA II - ZWEIPHASIGER NUTZUNGSZYKLUS

Das zweite Paradigma bedient das übergeordnete Leitprinzip *vom Programmieren zum Spezifizieren*, mit der die Modellierung, Ausführung und Änderung von wissensbasierten Software-Anwendungen in einen Nutzungszyklus des Software-Frameworks gebracht wird. Das Leitprinzip hat zum Ziel, die gesamte Entwicklung einer wissensbasierten Software-Anwendung ausschließlich und lediglich durch das Bereitstellen der domänen-bezogenen Wissensinhalte (i. e. klinisch-therapeutischer Prozess) zu vollziehen und somit die inhärente Trennung wissensverarbeitender Systeme von Daten, Wissen und Problemlösungsstrategie (Abbildung 3.1, Seite 29) zu beherzigen. Je weniger Programmieraufwand nötig ist, umso effizienter ist die Erzeugung und Pflege einer wissensbasierten Software-Anwendung. Spezifizieren meint hier die Beantwortung der Frage „Was soll die wissensbasierte Software-Anwendung aus klinischer Sicht leisten?" im Sinne eines Pflichtenheftes.

Damit werden folglich die Experten, also Ärzte und Pflegepersonal, direkt mit in den Entwicklungsprozess einbezogen und es entsteht die fortlaufende, enge Kooperation zwischen ihnen und den Entwicklern. In der Praxis bedeutet dies aber auch die Herausforderung, solch eine Verbindung hinsichtlich Verfügbarkeit, Finanzierung, Organisation, Arbeitsmodus, Evidenz, Rechte, etc. adäquat zu gestalten.

Mit dem ersten Paradigma der zwei Freiheitsgrade ist sichergestellt, dass das Software-Framework strukturell in der Lage ist, Multiplizität zu gewährleisten. Die funktionale Sicht auf das Software-Framework wird mit dem Paradigma des

zweiphasigen Nutzungszyklus eröffnet, der festlegt, wie neue und/oder zu modifizierende wissensbasierte Software-Anwendungen entstehen und im Zielsystem ausgeführt werden.

Abbildung 4.4 gibt den gesamten Nutzungszyklus grafisch wieder, der im Einzelnen wie folgt zu verstehen ist, wobei invariante Anteile wiederum grau hinterlegt sind.

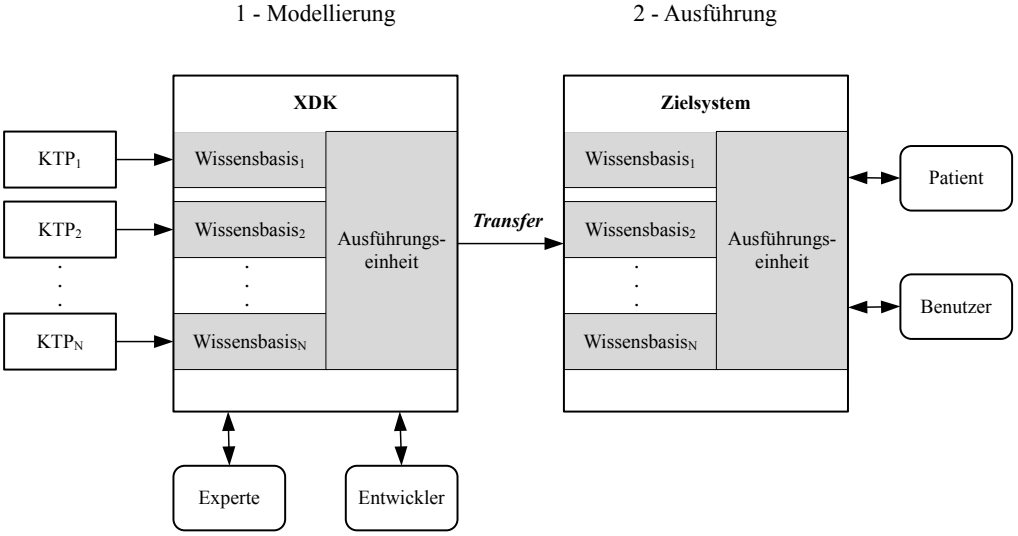

Abbildung 4.4 – *Modellierung, Transfer und Ausführung einer WBSA*

Phase 1 - Modellierung. In dieser Phase findet der Wissenserwerb und die Wissensmodellierung statt, d. h. ein oder mehrere klinisch-therapeutische Prozesse (KTP) werden erfasst und computerisiert. Dazu nutzen Experte und Entwickler eine integrierte Expertensystem-Entwicklungsumgebung (engl. Expert System Development Kit (XDK)). Das XDK entspricht einem Autorensystem, wie wir es in Abschnitt 1.3 bereits kennengelernt haben. Dort eingebettet ist eine *Ausführungseinheit*, mit der die aus den klinisch-therapeutischen Prozessen abgeleiteten Wissensbasen zum Leben erweckt werden können. Moderne XDK bieten darüber hinaus sämtliche Funktionalitäten, wie wir sie von Software-Entwicklungsumgebungen kennen, beispielsweise Code-Editor, Compiler, Interpreter, Debugger, Build-Automatisierung, Test-Frameworks, Projektmanagement und insbesondere Plugins und Erweiterungen.

Phase 2 - Ausführung. Nachdem Experte und Entwickler einen stabilen Stand der Modellierung eines klinisch-therapeutischen Prozesses erreicht haben, kann die korrespondierende Wissensbasis mitsamt der Ausführungseinheit aus dem XDK in das designierte Zielsystem exportiert werden. Dabei handelt es sich wohlgemerkt nicht um eine automatisierte Code-Generierung, sondern vielmehr um die physikalische Kopie der jeweiligen Quelltexte, die sowohl für Wissensbasen als auch für Ausführungseinheit identisch sind und bleiben. Nach erfolgtem Export kann das Zielsystem die neue oder geänderte wissensbasierte Software-Anwendung ausführen, die mit dem Benutzer, dem Zielsystem und dem Patienten interagiert.

Das Paradigma des zweiphasigen Nutzungszyklus realisiert vornehmlich die

Software-Qualitätsattribute Gebrauchstauglichkeit, Erweiterbarkeit und Wartbarkeit. Die Effizienz des Paradigmas wird entscheidend bestimmt mit dem Grad der invarianten gegenüber den variablen Anteilen: je mehr Anteile invariant bleiben können, desto effizienter der Nutzungszyklus.

Wie zuvor bereits angedeutet, kommt als XDK für die Phase der Modellierung ein Knowledge Wiki zum Einsatz, siehe Abschnitt 3.5.

4.6 PARADIGMA III - HYBRIDES VORGEHENSMODELL

Das Software-Framework wie auch damit entwickelte wissensbasiertes Software-Anwendungen sind für den Einsatz im medizinischen Umfeld vorgesehen. Derartige Produkte sind typischerweise als Medizinprodukt klassifiziert, für deren Entwicklung, Inverkehrbringen und Betrieb spezifische, externe Normen zu befolgen sind. Die Einhaltung dieser externen Normen wird periodisch und fortlaufend, behördlich überwacht und auditiert, beispielsweise vom Bundesinstitut für Arzneimittel und Medizinprodukte (BfArM) in Deutschland.

Für die Entwicklung von sogenannter medizinischer (Geräte-)Software ist die IEC 62304 Medizinische Geräte-Software - Software-Lebenszyklus-Prozesse [110] bindend. Sie stellt Mindestanforderungen an den Software-Lebenszyklus und fokussiert dabei auf einen prozessorientierten Ansatz. Eng verknüpft mit einem Entwicklungsprozess ist immer auch ein damit assoziiertes Vorgehensmodell, dass aus Sicht des Projektmanagements einen reibungslosen und effektiven Durchlauf des Entwicklungsprozesses unterstützt. Aus der klassischen Produktentwicklung sind u. a. das Wasserfallmodell, das Spiralmodell, das weit verbreitete V-Modell bekannt und seit kurzem moderne Ausprägungen wie agile Methoden, Scrum, Kanban etc.

Im vorliegenden Fall besteht eine zusätzliche Herausforderung bei der Wahl des geeigneten Vorgehensmodells und Entwicklungsprozesses darin, die Koexistenz konventioneller Software-Komponenten mit wissensbasierten Komponenten plausibel, lückenlos und präzise zu bewerkstelligen. Es gilt also, Prinzipien und Methoden aus dem Software Engineering mit denen des Knowledge Engineering zu fusionieren, um diese Heterogenität adäquat abbilden zu können.

Software Engineering. Um die Jahrtausendwende hielten objektorientierte Methoden immer mehr Einzug in die Software-Entwicklung. Im hochspezialisierten Bereich der eingebetteten Echtzeitsysteme (engl. Embedded Realtime Systems), zu dem auch Medizingeräte gehören, vollzog sich dieser Paradigmenwechsel erwartungsgemäß deutlich langsamer. Dies betraf vor allem Vorgehensmodelle und Entwicklungsprozesse. Gesucht für eine zukunftssichere Entwicklung und Pflege des Software-Frameworks wurde ein Software-Entwicklungsprozess, der Objektorientierung unterstützt, tauglich für die Entwicklung eingebetteter Echtzeitsysteme ist und gleichzeitig ermöglicht, mit anderen Vorgehensmodellen zu kooperieren.

Mit dem von Awad und seinem Team bei Nokia 1996 erschaffenen Octopus-Modell [36] wurden wir fündig. Octopus, später erweitert zu Octopus/UML,

beschreibt umfassend die objektorientierte Software-Entwicklung für mobile Geräte mitsamt eines inkrementell, evolutionären Lebenszyklus. Herauszustellen ist, dass Octopus die Entwicklung von vertikalen Subsystemen forciert und mit einem speziellen, vordefinierten Subsystem, dem sogenannten *Environment Wrapper*, die Einbindung etwaiger Hardware ermöglicht. Außerdem verfolgt das Octopus-Modell konsequent die aus der Systemtheorie bekannte Dimensionierung eines Systems in Struktur, Funktion und Dynamik.

Bezogen auf die Basis-Systemarchitektur (Abbildung 4.2) des Software-Frameworks korrelieren die Subsysteme zu den Slots und der Environment Wrapper zum Slot *Zielsystem*. Die Nutzung einer schichtenbasierten Architektur (Anwendungsschicht, Environment Wrapper, Physikalische Schicht) kommt unserem Schichtenmodell nahe und das schrittweise Aufwachsen des zu entwickelnden Systems in inkrementellen, evolutionären Schritten begünstigt die Forderung nach Flexibilität und Kundenzentrierung. Damit ist Octopus/UML die ideale Kombination von Vorgehensmodell und Entwicklungsprozess für die konventionellen Software-Komponenten des Software-Framework.

Knowledge Engineering. Für die Entwicklung der wissensbasierten Anteile des Software-Frameworks sind die eher technisch ausgelegten Methoden des konventionellen Software Engineering weniger geeignet. Hier braucht es Unterstützung auf abstrakter Ebene, um die interdisziplinäre Zusammenarbeit zwischen Domänenexperte und Wissensingenieur zu erleichtern. Die Disziplin des *Knowledge Engineering* widmet sich diesen besonderen Anforderungen. Wir definieren sie wie folgt:

Definition KNOWLEDGE ENGINEERING

> Knowledge Engineering, zu deutsch häufig: *Wissensverarbeitung*, *Wissensmodellierung*, ist ein Bereich der künstlichen Intelligenz, der sich mit der Entwicklung von Systemen befasst, die menschliches Wissen aufnehmen und anwenden, um Probleme zu lösen oder Entscheidungen zu treffen, die normalerweise menschliche Intelligenz erfordern würden.

Mit Aufkommen und Weiterentwicklung des Knowledge Engineering hat sich auch das Berufsbild des Wissensingenieurs (engl. Knowledge Engineer) etabliert. Im Knowledge Engineering sind die vier wesentlichen Arbeitsschritte auf dem Weg zu einem wissensbasierten System:

1. **Wissenserfassung.** Gewinnung von Wissen von Experten oder aus Daten
2. **Wissensmodellierung.** Strukturierung des Wissens in verständlichen Modellen
3. **Wissensrepräsentation.** Speichern des Wissens in formalen Strukturen wie Regeln oder semantischen Netzen
4. **Wissensnutzung.** Implementierung von Wissen in Systemen für spezifische Anwendungen

Es gibt zwei Vorgehensmodelle, die im Knowledge Engineering weite Verbreitung gefunden haben:

MIKE. *Model-based and Incremental Knowledge Engineering* [76] ist ein systematischer Ansatz zur Entwicklung wissensbasierter Systeme. Die Methodik und das Vorgehensmodell kombinieren modellbasierte und inkrementelle Techniken, um die Strukturierung und Implementierung von Wissenssystemen zu unterstützen. MIKE wurde entwickelt, um die Komplexität und Herausforderungen bei der Erstellung solcher Systeme zu bewältigen, indem es auf eine klare Modellierung und schrittweise Entwicklung setzt.

CommonKADS. *Common Knowledge Acquisition and Design System* [30] ist eine Methodik zur Entwicklung wissensbasierter Systeme und eine der am weitesten verbreiteten und anerkannten Vorgehensmodelle in diesem Bereich. Es handelt sich um eine strukturierte Zusammenstellung von Konzepten, Modellen und Techniken, die den gesamten Entwicklungsprozess unterstützen, von der Analyse und Erfassung des Wissens bis hin zur Implementierung und Wartung. Während der Anwendung von CommonKADS entstehen Modelle für Organisation, Aufgaben, Agenten, Wissen, Kommunikation und Entwurf.

Aus diesen beiden Modellen, einigen Kernprinzipien des Octopus/UML, aber auch von anderen Ansätzen und Perspektiven inspiriert, haben wir für die speziellen Anforderungen des Software-Frameworks ein eigenes Vorgehensmodell erstellt.

Clinical Knowledge Engineering Process (CKEP). Das aus dieser Fusion hervorgebrachte Vorgehensmodell CKEP [5] gibt acht sequentielle Phasen vor, die je nach Bedarf optional und iterativ durchlaufen werden können. Sie sind zur besseren Orientierung auf die bekannten Standardphasen *Initiierung, Analyse, Design & Implementierung, Qualifikation & Freigabe* und *Betrieb* abgebildet.

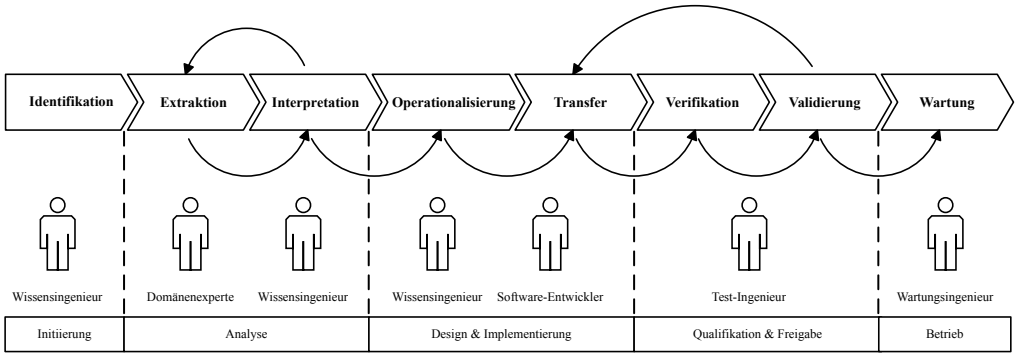

Abbildung 4.5 – *Vorgehen beim Clinical Knowledge Engineering*

Der Gesamtzusammenhang, insbesondere die Iterationsmöglichkeiten zwischen den Phasen, ist in Abbildung 4.5 illustriert. Die einzelnen Phasen sind:

Identifikation. Ein Entwicklungsprojekt soll gestartet werden, die Zweckbestimmung des Produktes ist bekannt. In der Identifikationsphase werden nun die Vor- und Rahmenbedingungen für die Durchführung des Vorhabens geschaffen. Der avisierte Wissensbereich - die Domäne - wird näher untersucht, um zu entscheiden, ob Umfang und Qualität des Expertenwissens adäquat und ausreichend sind. Dazu gehört zwingend die Klärung der Evidenz des Wissens, siehe Abschnitt 2.1.2. In welcher Form liegt das Wissen vor? Stammt es von einem Individuum und

liegt nur narrativ vor? Oder gibt es bereits eine Sx-Leitlinie (Abschnitt 2.1.3) eines internationalen Gremiums? Gibt es wiederverwendbare Materialien? Stehen etwaige Schutzrechte im Wege? Wie kann die Zusammenarbeit mit den Experten organisiert, finanziert und vertraglich ausgestaltet werden? Konnten diese wichtigen Fragen positiv beantwortet werden, beginnt die Initiierung des Projektes mit Bereitstellung der Betriebsmittel, Aufbau eines Teams und Erstellung einer ersten Zeit- und Ressourcenplanung.

Neben dem Wissensingenieur ist hier ganz entscheidend der benannte Projektleiter involviert.

Extraktion. Im Englischen auch *Elicitation* genannt. In dieser Phase werden die relevanten Wissensinhalte zusammengetragen und in einem erstem Modell, dem Extraktionsmodell, formal gesammelt. Analog zu den Software Requirements Specification des Octopus/UML entstehen so die Clinical Requirements Specification (CRS). Außerdem erfolgt eine initiale Bewertung der klinischen Risiken, die von einem späteren klinisch-therapeutischen Prozess bzw. dessen wissensbasierte Software-Anwendung ausgehen können. Schließlich wird untersucht, wie das extrahierte Wissen klinisch evaluiert werden kann und dazu eine entsprechende Strategie erarbeitet. Diese Aktivität ist essenziell und entscheidet über Fortführung des Projektes und zukünftigen wirtschaftlichen Erfolg des Produktes. Ist eine Evaluierung via Literaturstudium ausreichend, so ist dies wesentlich kostengünstiger als die Durchführung einer lang angelegten, teuren und aufwändigen klinischen Studie. Die Extraktionsphase kann auch simultan mit der Interpretationsphase erfolgen, was in der Praxis häufig passiert.

Domänenexperten und Wissensingenieur arbeiten hier sehr eng zusammen.

Interpretation. Die Inhalte des Extraktionsmodells sollen in der Interpretationsphase grafisch dargestellt und prozessual zu einem Flussdiagramm zusammengeführt werden. Dies können einfache Programmablaufpläne sein, proprietäre oder fortgeschrittenere Notationen wie UML, SysML etc. Die dabei entstehende Clinical Workflow Specification (CWS) bildet zusammen mit der CRS die vollständige Spezifikation der WBSA, genauer: der Anwendungsschicht im Slot XPS oder: das Lastenheft. Dem Modularisierungsgedanken folgend, werden wiederverwendbare Fragmente eingebunden bzw. wieder zu verwendende entsprechend modelliert. Es bilden sich bereits in dieser Phase erste Testspezifikationen (Test-First-Ansatz), zudem wird die Möglichkeit nach eigenen Schutzrechten beleuchtet.

Auch hier arbeiten Experte(n) und Wissensingenieur (noch) eng zusammen.

Operationalisierung. Das zentrale Arbeitsergebnis der Operationalisierungsphase ist die mit dem jeweiligen XDK erstellte Wissensbasis, also die computerisierte, ablauffähige Fassung des klinisch-therapeutischen Prozesses, wie sie sich in der Nutzungsphase 1 *Modellierung* herausbildet, siehe Abbildung 4.4. Der Wissensingenieur führt zusammen mit dem Software-Entwickler eine nach Kodierungskonventionen geleitete Inspektion der Wissensbasis durch. Für die später folgende Verifikationsphase werden Strategie und Testspezifikation erstellt und weiter verfeinert. IT-Werkzeuge und Vorlagen können dabei mitentwickelt werden.

In dieser Phase arbeiten nur noch Wissensingenieur und Software-Entwickler zusammen. Es findet folglich ab dieser Phase eine Übergabe an die interne

Entwicklungsabteilung statt, nach dem in den vorangehenden Phasen viel Vorort bei den Experten, sprich: im klinischen Umfeld, gearbeitet wurde.

Transfer. Mit der Transferphase vollzieht sich der Übertrag der Wissensbasis in das designierte Zielsystem, so wie er in der Nutzungsphase als *Transfer* vorgesehen ist, siehe Abbildung 4.4. Damit ist das bis dato entwickelte Produkt für die Nutzungsphase 2 *Ausführung* bereit, wobei Integrationstest die Fehlerfreiheit des Transfer dokumentieren.

Der Software-Entwickler kümmert sich vorrangig um das Abwickeln der Transferphase.

Verifikation. Ziel der Verifikationsphase ist die formale Beantwortung der Frage „Haben wir die Wissensbasis korrekt gebaut?" Die für diesen Nachweis zuvor spezifizierten Testfälle werden durchlaufen, deren Ergebnisse dokumentiert, etwaige Abweichungen bewertet und korrigiert, was zu einer erneuten Durchführung der Verifikation führen kann. Dieser Zyklus wiederholt sich solange, bis keinerlei Abweichungen mehr auftreten.

Validierung. Ziel der Validierungsphase ist die formale Beantwortung der Frage „Haben wir die korrekte Wissensbasis gebaut?" Die Antwort darauf findet sich beim Abgleich der Wissensbasis mit den Inhalten von CRS und CWS, deren Inhalte wiederum die Erstellung der Validierungsspezifikation leiten. Zusätzlich soll in dieser Phase die zuvor strategisch definierte klinische Evaluierung durchgeführt und erfolgreich abgeschlossen werden, bevor sich eine formale Freigabe des Gesamtsystems und damit der Abschluss des Entwicklungsprojektes anschließen kann.

Für die Verifikations- und Validierungsaktivitäten ist der Testingenieur bzw. eine Gruppe davon verantwortlich.

Wartung. Das entwickelte und freigegebene Produkt (i. e. wissensbasierte Software-Anwendung) wurde in den Markt gebracht und befindet sich im klinischen Einsatz. Während der Wartungsphase, auch Lifecycle Management genannt, werden Kundenrückmeldungen gesammelt, bearbeitet und Änderungsanträge umgesetzt.

Dies übernimmt der Wartungsingenieur mit seinem Team.

Insgesamt lässt sich sagen, dass der CKEP iterativ-inkrementell-evolutionäres Vorgehen bei der Entwicklung wissensverarbeitender Systeme adäquat ermöglicht. Er lässt sich daher gut mit dem Octopus/UML in Einklang bringen, begünstigt agile Methoden des Software Engineering und erleichtert vor allem die anspruchsvolle Zusammenarbeit zwischen Experte und Wissensingenieur. Mit erneutem Blick auf die Basis-Systemarchitektur (Abbildung 4.2) kann festgestellt werden, dass sich der CKEP ausschließlich um die Belange des XPS-Slots $WB_{1...N}$ kümmert. Alle anderen Komponenten deckt der Octopus/UML ab.

Das dritte Paradigma des hybriden Vorgehensmodells realisiert die Software-Qualitätsattribute Erweiterbarkeit, Interoperabilität und Robustheit.

Wir haben in diesem Kapitel die spezifische Methodik zur Lösung der Problemstellung dargelegt, indem wir zunächst die Anforderungen an das Software-Framework

festgelegt haben, bevor eine flexible Systemarchitektur erarbeitet wurde. Letztere bestimmt auch, wie die Dynamik des Gesamtsystems gehandhabt wird.

Die aus den Systemanforderungen abgeleiteten drei Paradigmen sind entscheidend für eine zielführende Konstruktion des Software-Frameworks, die Akzeptanz, Erweiterbarkeit, eine hohe Integrationsfähigkeit und damit auch die gewünschte Zukunftssicherheit gewährleistet.

Leitprinzip und Design-Prinzip fanden während der Methodik stets Berücksichtigung, was sich im weiteren Verlauf der Arbeit fortsetzen wird. Das gilt ebenfalls für die drei Paradigmen.

KAPITEL 5

TECHNOLOGIE UND IMPLEMENTIERUNG

Die im vorangehenden Kapitel vorgestellte Methodik mit ihren Paradigmen, den Anwendungsfällen, der Systemarchitektur und der Ablaufdynamik stellen die Basis für die Implementierung der Gesamtlösung und leiten diese.

Als integrierte Entwicklungsumgebung für den wissensverarbeitenden Teil der Gesamtlösung kommt ein Knowledge Wiki zum Einsatz, das für die speziellen Systemanforderungen modifiziert und ergänzt wurde. Wir beschreiben das Knowledge Wiki sehr detailliert, da es das zentrale Instrument im Knowledge Engineering ist. Insbesondere die grafische Modellierung prozessualer Abläufe und das Formalisieren zeitlicher Abläufe und Abhängigkeiten stehen dabei im Mittelpunkt.

Anschließend legen wir dar, wie ein als Wissensbasis in der Entwicklungsumgebung modellierter Prozess in ein Zielsystem gelangt und innerhalb welchen Lebenszykluses dies geschieht.

Nach der Erläuterung der für das Laufzeitverhalten zuständigen Anwendungsteuerung des Software-Frameworks und des dafür verwendeten Datenmodells endet diese Kapitel mit der Antwort auf die wichtige Frage, wie dedizierte Zielsysteme für die Ausführung klinisch-therapeutischer Prozesse technologisch anzubinden sind.

5.1 EINORDNUNG IN FORSCHUNG UND TECHNIK

Die Ermittlung verwandter Arbeiten und Ansätze zu Software-Technologien, die im Bereich Medizintechnik Anwendung fanden bzw. finden, gestaltet sich schwierig und liefert kaum wissenschaftlich adäquate Literaturstellen. Für gewöhnlich werden Implementierungsdetails in diesem Anwendungsbereich strikt vertraulich gehalten oder unterliegen gar lizenz- oder patentrechtlichen Bestimmungen. Wir fokussieren daher an dieser Stelle auf Aspekte der Neuartigkeit und auf wissenschaftliche Beiträge, die für diese Arbeit erbracht wurden.

Ebenso wie die im vorangehenden Kapitel vorgestellte, spezifische Methodik des Software-Frameworks (SWFW) waren auch für Auswahl und Anwendung der Implementierungstechnologien im Wesentlichen die zwei Prinzipien und drei Paradigmen (siehe Abschnitt 1.4) die zentralen Stimuli. Dies gilt im besonderen Maße für das Leitprinzip „Vom Programmieren zum Spezifizieren" sowie für das Design-Prinzip „Trenne invariante von variablen Anteilen". Damit gelangten wir zur objekt-orientierten Programmiersprache Java mit den verbundenen Java-Technologien, die unserem Software-Framework sowie eine hohe Portier- und Erweiterbarkeit verschafften. Entscheidend war jedoch auch die in der Methodik getroffene Festlegung auf wissensverarbeitende und Expertensysteme, die sich ja ebenfalls durch ausgeprägte Objektorientierung auszeichnen.

Der Zeitraum der Neuentwicklung des Software-Frameworks lässt sich auf die Jahre 2000 bis 2003 eingrenzen. Zu dieser Zeit waren Medizingeräte - insbesondere bei Dräger - nahezu ausschließlich als eingebettete Echtzeitsysteme (engl. Embedded Realtime Systems) auf Micro-Controller Basis ausgeführt, die in den Hochsprachen C oder seltener C++ programmiert wurden. Die Verwendung von Java war - zumindest im Hause Dräger - neu und erstmalig. Gleiches trifft auch auf den Einsatz von Expertensystemen für Medizinprodukte zu, und deren autonome Verstellung von Einstellwerten.

Außerdem vorteilhaft war die Tatsache, dass mit dem Sprachstandard Java 2 Platform, Standard Edition (J2SE) bereits verschiedene Laufzeitumgebungen wie beispielsweise JWorks der Firma WindRiver für das Echtzeitbetriebssystem VxWorks [123] verfügbar wurden, mit denen der volle Java-Sprachumfang der Version 2.0, wie von Bloch [32] beschrieben, nutzbar waren. Die Verwendung von JWorks für eingebette Systeme bei Dräger war ebenfalls neuartig. Mit der Migration des Sprachstandards auf die Version Java 5.0, eng orientiert an den Ausführungen in Chandrasekhara et al. [47], endeten im Jahre 2008 die Anpassungen an die Evolution der Programmiersprache Java.

Zu den Java-Technologien zählen darüber hinaus auch solche Frameworks zur Erzeugung von graphischen Benutzeroberflächen (GUI). Unser Software-Framework startete mit der Bereitstellung des *Abstract Window Toolkit (AWT)*, später abgelöst durch *Swing*, für wissensbasierte Software-Anwendungen, die lokal im Medizinprodukt integriert sind und erweiterte sich um Google's Web Toolkit (GWT) für Anwendungen, die web-basiert zur Ausführung kommen sollen. Allen GUI-Frameworks gemein ist, dass sie im Rahmen des Software-Frameworks erstmalig bei Dräger Verwendung fanden. Eine weitere Java-Technologie, die sich direkt aus der basalen Anforderung mit nicht-Java basierten Technologien zu

kooperieren (siehe oben) ergab, ist das Java Native Interface (JNI) [43]. Diese Brückenschnittstelle benutzten wir sowohl zur datentechnischen Anbindung an Zielsysteme als auch zur autonomen Steuerung von Medizingeräten, beides ebenfalls erstmalig bei Dräger.

Die zentrale Komponente zur Implementierung des Leitprinzips „Vom Programmieren zum Spezifizieren" ist das Knowledge Wiki *KnowWE*, das zusammen mit einem Autorensystem, einer Ausführungseinheit (d3web) und einer Vielzahl von Zusatzfunktionen das Expert System Development Kit (XDK) bildet, vergleichbar mit einem modernen Software Development Kit (SDR). Der Einsatz solch einer Technologie bei Dräger ist neu und nach Auskunft der Betreiber von d3web im kommerziellen Bereich nicht existent, wogegen einige akademische Anwendungen mit d3web entwickelt wurden, wie die Web-Seite der Universität Würzburg darlegt [121].

Schließlich und ebenfalls neuartig ist die Implementierung der Fähigkeit, multiple Wissensbasen zu bedienen, die auch dynamisch zur Laufzeit einer wissensbasierten Software-Anwendung wahlweise gewechselt werden können. Diese funktionale Eigenschaft widmet sich der Umsetzung des Paradigmas „Zwei Freiheitsgrade" und geht zurück auf Abbildung 3.1(c).

5.2 ENTWICKLUNGSUMGEBUNG

5.2.1 MODELLIERUNG IM AUTORENSYSTEM

Effizienz und Zukunftssicherheit des gesamten Software-Frameworks ergeben sich unmittelbar aus den in Kapitel 4 vorgestellten Paradigmen. Komfort in der Bedienung und Flexibilität bezüglich Änderbarkeit spielen insbesondere für die wissensverarbeitenden Komponenten eine übergeordnete Rolle. Diesen muss das Expert System Development Kit mitsamt seinem Autorensystem gerecht werden.

KnowWE (Knowledge Wiki Environment) ist ein semantisches Wiki, das speziell für die kollaborative Erfassung, Pflege und Anwendung von Wissen entwickelt wurde. Es ermöglicht Nutzern, formale Wissensstrukturen - etwa in Form von Ontologien unter Verwendung von Standards wie der Web Ontology Language (OWL) - mit informellen, beschreibenden Inhalten zu verknüpfen. Dadurch können komplexe Wissensbasen, wie beispielsweise Entscheidungsunterstützungssysteme oder diagnostische Systeme, gemeinschaftlich erstellt und kontinuierlich verbessert werden. Das System unterstützt dabei interaktiv, kooperative Abläufe, in denen Experten problembezogenes Wissen einbringen, verfeinern und anwenden können. Ferner bietet KnowWE Funktionalitäten zur Erleichterung der Entwicklungstätigkeiten wie beispielsweise Continuous Integration, Debugging, Dashboards, Aufgabenmanagement, Kodierkonventionen und weitere.

KnowWE ist ein Teilprojekt der d3web-Familie, die im Jahre 2004 um Prof. Puppe an der Universität Würzburg entstanden ist [31]. Einen umfassenden Einblick in Grundideen, Konzepte, Funktionsweise und Anwendung von KnowWE geben Baumeister et al. [77]. Spezifische Anwendungen finden sich bei Hatko et al. für den Bereich Diagnostik [107] sowie für den Bereich der medizinischen Leitlinen [18].

Während KnowWE das Autorensystem liefert, stellt d3web die Ausführungseinheit inklusive der Inferenzmaschine samt unterschiedlicher Problemlösungsstrategien wie regelbasiertes, probabilistisches, nicht-monotones und/oder temporales Schließen bereit, siehe Abschnitt 3.4. Die praktische Arbeit mit KnowWE ähnelt sehr den bekannten Wikis wie beispielsweise Wikipedia, nur eben ergänzt um Formalismen zur Modellierung von Problemlösungs- und fallspezifischem Wissen.

In Abbildung 5.1 ist dargestellt, wie sich KnowWE dem Anwender präsentiert. Dargestellt ist die Einstiegsseite der Wissensbasis aus unserem Beispiel zur Sepsiserkennung mit erläuternden Informationen im rechten Teil sowie den inhaltlichen Strukturelementen auf der linken Seite als Navigationsmenü. Die Gestaltung des linksseitigen Menüs ist vereinheitlicht für jeden mit KnowWE zu modellierenden klinisch-therapeutischen Prozess und führt über den Punkt *Clinical Workflow* in die eigentliche Prozessdarstellung.

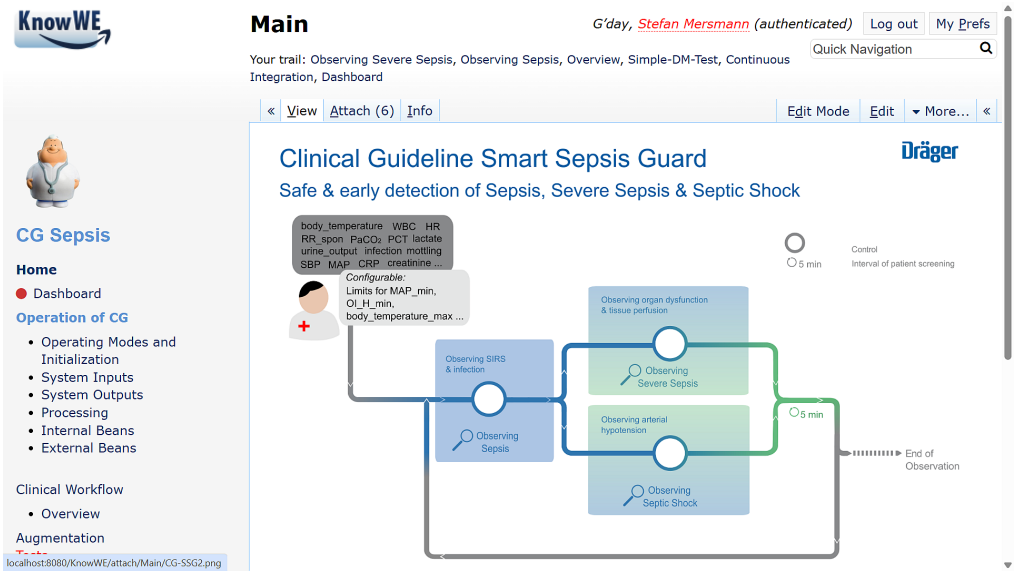

ABBILDUNG 5.1 – *Einstiegsseite von KnowWE*

Die Ausführung eines Falles mit einer Wissensbasis vollzieht sich in KnowWE mit Hilfe des d3web-Kerns als sogenanntes *Interview*, bei dem der Benutzer sukzessive Eingaben tätigt, die unmittelbar von der Problemlösungskomponente verarbeitet werden und schließlich zu einer Lösung bzw. Entscheidung führen.

Ein Beispiel eines Interview-Ausschnittes findet sich in Abbildung 5.2, ebenfalls für unser Anwendungsbeispiel Sepsiserkennung. In der fertiggestellten wissensbasierten Software-Anwendung übernimmt dann das jeweilige Zielsystem die Abarbeitung des Interviews, an Stelle des Benutzers.

KnowWE baut auf der Open-Source Wiki-Engine JSPWiki auf und verwendet JavaServer Pages (JSP) für seine Implementierung. Es ist flexibel und kann auf verschiedenen Betriebssystemen und mit verschiedenen Webservern betrieben werden, solange eine entsprechende Java-Umgebung vorhanden ist. Weiterhin unterstützt JSPWiki Plugins und Erweiterungen, die es ermöglichen, einfach

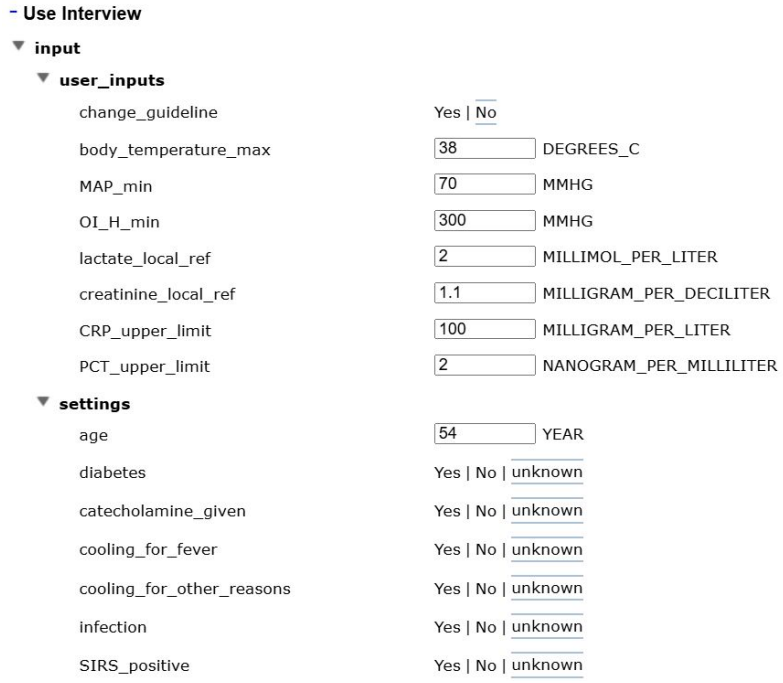

Abbildung 5.2 – *Dialogformular eines KnowWE-Interviews*

und komfortabel zusätzliche Funktionalitäten hinzuzufügen, wie etwa erweiterte Textformatierungen, Diagrammerstellung oder Datenbankanbindung.

Der entscheidende Faktor für die Wahl von JSPwiki ist neben der freien Verfügbarkeit die schier unbegrenzte Möglichkeit, Erweiterungen und/oder Modifikationen Java-basiert zu implementieren und somit quasi ad libitum eine proprietäre Wiki-Umgebung zu erschaffen.

Neben den bereits erwähnten Vorteilen, eröffnet die Nutzung eines Knowledge Wiki für Entwicklungszwecke darüber hinaus die Möglichkeit, als singuläre Datenquelle zu fungieren. Dies bedeutet, dass alle für Entwicklung, Zulassung und Betrieb relevanten Ergebnisdokumente ausschließlich in KnowWE entstehen, abgelegt sind, geprüft und freigeben und schlussendlich ausgeliefert werden. KnowWE erspart somit die gleichzeitige Verwendung mehrerer, unterschiedlicher IT-Systeme für das Dokumentenmanagement und folgt damit den Prinzipen einer Single Source of Truth (SSOT).

5.2.2 GRAFISCHE MODELLIERUNG

Mit der Technologie der Knowledge Wikis ist der Grundstein für eine interdisziplinäre Zusammenarbeit zwischen Domänenexperte auf der einen Seite sowie Wissensingenieur und Software-Entwickler auf der anderen Seite gelegt. Eine weitere konsequente Umsetzung des Leitprinzips *vom Programmieren zum Spezifizieren* ist durch die grafische Notation *DiaFlux* entstanden. DiaFlux ist eine KnowWE-Erweiterung, die die grafische Modellierung von Prozessen im Allgemeinen und klinisch-therapeutischen Prozessen im Besonderen ermöglicht, mit dem

Ziel, die Wissenserfassung komfortabler und einfacher zu gestalten. Idealerweise sollen Domänenexperten in die Lage versetzt werden, anfänglich auch ohne Zutun eines Wissensingenieurs ihre Prozesse zu computerisieren.

DiaFlux ist ein wesentliches Arbeitsergebnis des BMBF-Forschungsprojektes WiM-Vent (Wissens- und modellbasierte Therapiesteuerung in der Medizin) und wurde 2011 von Hatko et al. [105] der wissenschaftlichen Welt erstmals vorgestellt. Darin sind folgende basale Zielsetzungen genannt:

- Wiederholte Ausführung von Teilprozessen (Modularisierung)
- Repräsentation von Zeit inklusive temporalem Schließen
- Nicht-monotones Schließen, i. e. Truth Maintenance System (TMS)
- Parallelität interprozessualer Abläufe
- Modularität im Sinne von Wiederverwendung
- Testbarkeit (Validierung, Evaluierung)

Für die grafische Prozessmodellierung stellt DiaFlux die in Tabelle 5.1 aufgelisteten Modellierungselemente zur Verfügung.

Element	**Semantik**
Start	Markierung des Einstiegs in einen (Sub)Prozess
Ende	(Sub)Prozess endet in diesem Element
Frage	Dateneingabe während eines Interview
Modul	Definition eines Subprozesses
Abstraktion	Berechnungen, Qualifikatoren, Synthese
Entscheidung	Logische Verzweigung
Lösung	Schlußfolgerung oder Ausschluß einer Diagnose/Lösung
Pause	Aussetzen der Ausführung für eine vorgegebene Zeit
Snapshot	Verhindert Rücknahme von Schlussfolgerungen vor dieser Marke
Kommentar	Dokumentation ohne Einfluss auf Ausführung
Test	Ausführung einer vorgegebenen Testaktivität

TABELLE 5.1 – *Modellierungselemente in DiaFlux*

Abbildung 5.3 zeigt einige der Modellierungselemente von DiaFlux in einem Beispiel aus der S3-Leitlinie zur Erkennung und zur Behandlung septischer Erkrankungen (Abschnitt 2.1.3). Modelliert ist der Subprozess zur synthetisierenden Unterscheidung zwischen den drei Ausprägungen einer Sepsis. Darin finden sich

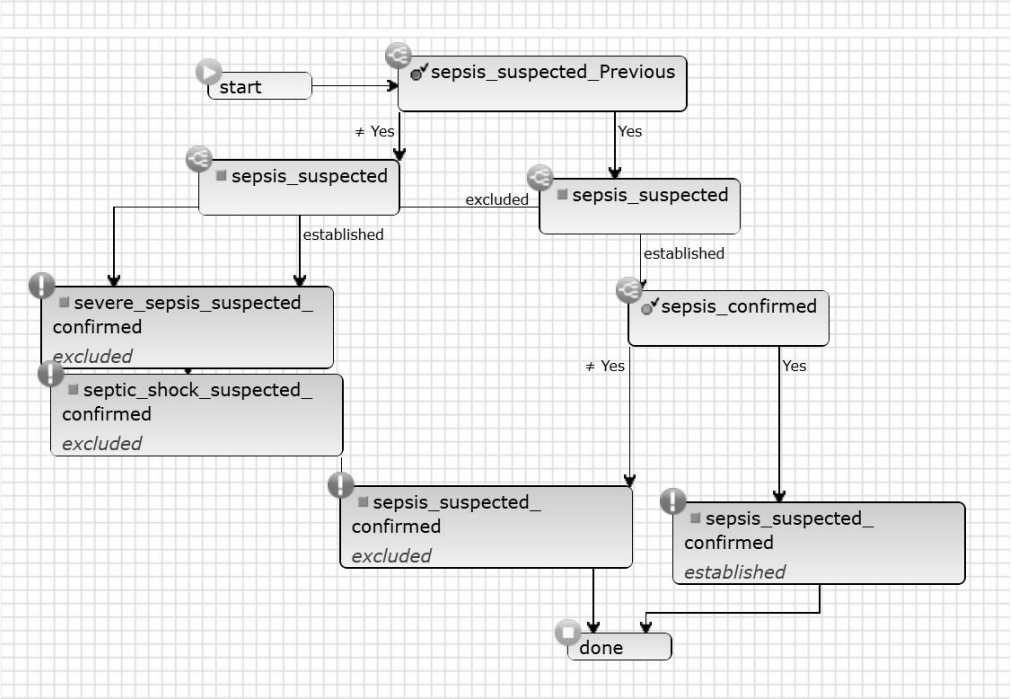

Abbildung 5.3 – *Grafische Prozessmodellierung in KnowWE*

neben Start- und Endeknoten Entscheidungsknoten (*sepsis_suspected*) sowie Lösungselemente zur Diagnosebildung (*sepsis_suspected, severe_sepsis_suspected, septic_shock_suspected*). Die Elemente selbst sind anhand von kleinen, runden Icons (oben links) identifizierbar.

Über die reine Modellierung von Prozessen mit DiaFlux hinaus gibt es zahlreiche weitere Funktionen zur Validierung, Visualisierung von Prozessdurchläufen, Berechnung von Metriken, Statistiken und Laufzeiten, Entdeckung von Anomalien etc. [19].

5.2.3 ZEITDATENBANK UND TEMPORALES SCHLIESSEN

Im Grunde genommen bildet ein klinisch-therapeutischer Prozess das kognitive Verhalten eines ärztlichen Behandlers zur Lösung einer spezifischen, medizinischen Fragestellung ab. Kennzeichnend und von besonderer Bedeutung für klinisch-therapeutische Prozesse sind zeitliche Zusammenhänge. So nutzen Ärzte und Pflegepersonal intensiv absolute und relative Zeitpunkte, zeitliche Verläufe und „Landmarken", die sie logisch miteinander verknüpfen, um zu Schlussfolgerungen, Diagnosen und therapeutischen Aktionen zu gelangen. Solche temporalen Betrachtungen sind stets retrospektiv und von der Bildung und Nutzung von Prognosen zu unterscheiden.

Typische Fragen, die ein Domänenexperte beantworten will sind beispielsweise:

- Wie hoch war die Körpertemperatur heute um 12:30 Uhr?
- Welchen Wert hatte der Beatmungsdruck zu Beginn der Therapie?
- Welchen Wert hatte die Sauerstoffsättigung als der Beatmungsdruck erstmalig über 18 mbar stieg?
- War die Herzfrequenz um 12:30 Uhr größer als zu Beginn der zweiten Medikamentengabe?
- Wie lange liegt die Diagnose „Zentrale Hyperventilation" schon an?
- Wann wurde die Therapiephase „Entwöhnung" verlassen?
- Gab es vor der Diagnose „Sepsis vermutet" einen Anstieg der Herzfrequenz um mindestens 30 %?
- Ist die Atemfrequenz in den letzten 20 min um mehr als 10 % gefallen?

Die Erweiterung *Time-DB* ergänzt die KnowWE-Funktionalitäten um genau diese Anforderungen und implementiert damit das in Abschnitt 3.4 vorgestellte temporale Schließen. Es werden alle Objektwerte einer Wissensbasis während der Ausführung mitsamt dem Zeitpunkt ihrer Entstehung und ihrer Änderungen aufgezeichnet und persistent gespeichert. Genauer gesagt: alle Werte der Modellierungselemente Frage, Abstraktion und Lösung. Technisch realisiert dies die in Java verfügbare Systemzeit `System.currentTimeMillis()`, die bei jedem Inferenzvorgang zusammen mit dem Wert eines Objektes abgelegt wird.

Zusätzlich zu den Operatoren der temporalen Intervalllogik von Allen [75] bietet die Time-DB noch weitere Verknüfungs- und Auswerteoptionen an, mit denen obige Fragen beantwortbar sind.

Innerhalb der Time-DB sind dazu Datentypen gemäß Tabelle 5.2 definiert.

Datentyp	**Semantik**
HISTORY_NUMBER	Zeitintervall von numerischen Objekten
HISTORY_STRING	Zeitintervall von Zeichenketten-Objekten
HISTORY_DURATION	Zeitintervall von Objekten des Typs Zeitdauer
HISTORY_DATE	Zeitintervall von Objekten des Typs Zeitpunkt
HISTORY_BOOLEAN	Zeitintervall von Objekten des Typs Boolean
DURATION	Einzelobjekt vom Typ Zeitdauer
DATE	Einzelobjekt vom Typ Zeitpunkt

TABELLE 5.2 – *Datentypen der Time-DB*

Die Funktionen der Time-DB, die wir in dieser Arbeit für das temporale Schließen genutzt haben, listet Tabelle 5.3.

Funktion	**Semantik**
[]()	Gesamter Inhalt der Time-DB für einen angeforderten Datentyp
average()	Durchschnitt von Werten eines Zeitintervalls
count()	Anzahl von Werten in einem Zeitintervall
delta()	Differenz zwischen Minimum und Maximum innerhalb eines Zeitintervalls
duration()	Zeitdauer von Werten eines Zeitintervalls
filter()	gemäß logischem Operator gefilterte Werte eines Zeitintervalls
firstChange()	Zeitpunkt der ersten Wertänderung in einem Zeitintervalls
gradient()	Richtung und Rate der größten Zu-/Abnahme eines Wertes über ein Zeitintervall
isNewer()	Ist ein Wert „jünger" als ein anderer Wert?
latestChange()	Zeitpunkt der letzten Änderungen eines Wertes
latestValue()	Letzter Wert aus ein oder mehreren Zeitintervallen
max()	Maximun von Werten eines Zeitintervalls
median()	Median von Werten eines Zeitintervalls
min()	Minimun von Werten eines Zeitintervalls
percentEqual()	Prozent der Zeit, in der ein Objekt einen bestimmten Wert in einem Zeitintervall hatte (weitere für größer/größer-gleich/kleiner/kleiner-gleich
since()	Zeitpunkt, ab dem ein Wert unverändert ist
subHistory()	Ausschnitt aus einem Zeitintervall
valueAt()	Wert eines Objektes zu einem bestimmten Zeitpunkt

Tabelle 5.3 – *Funktionen der Time-DB*

Sämtliche Funktionen der Time-DB lassen sich entweder direkt im Markup von KnowWE unterbringen (Algorithmus 5.1) oder grafisch in DiaFlux (Abbildung 5.4).

```
1
2  %%Rule
3  IF eval(gradient(measurement[-10s, 0s]) > 1)
4  THEN ...
5  %
6
7  %%Rule
8  IF measurement = known
9  THEN myGradient = eval(gradient(measurement[-10s, 0s]))
10 %
11
12 %%Variable myGradient = gradient(measurement[-10s,0s])
```

ALGORITHMUS 5.1 – *Anwendung der Time-DB im KnowWE-Markup*

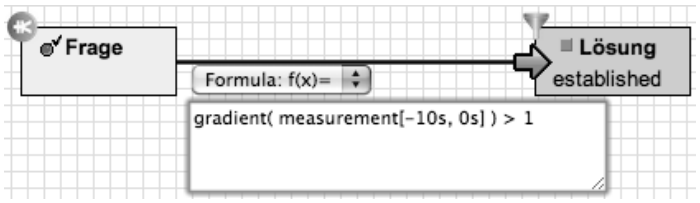

ABBILDUNG 5.4 – *Anwendung der Time-DB mit DiaFlux*

Das KnowWE-Plugin *Time-DB* ist ausschließlich als kommerzialisierte Version erhältlich.

5.3 TRANSFER IN EIN ZIELSYSTEM

Mit der Nutzung von KnowWE als Expert System Development Kit (XDK), inklusive dessen spezifische Erweiterungen, Anpassungen und Konfigurationen, ist die Modellierung klinisch-therapeutischer Prozesse und damit die erste Phase *Modellierung* des zweiphasigen Nutzungszyklus (Abschnitt 4.5) realisiert. Auch das Paradigma der zwei Freiheitsgrade (Abbildung 4.3, Seite 48) ist bezüglich multipler klinisch-therapeutischer Prozesse erfüllt, da mit KnowWE weitere Wissensbasen erstellt und eingebunden werden können.

Voraussetzung für die zweite Phase *Ausführung* des Nutzungszyklus ist der Transfer aller benötigten Komponenten in das jeweilige Zielsystem. Hierbei ist es unerheblich, um welches Zielsystem es sich handelt, solange selbiges die Migrationsvoraussetzungen erfüllt, von denen die wichtigste ein kompatibles Java-Laufzeitsystem ist.

Der Transfer ist ein invarianter Prozess, bei dem vordefinierte Software-Komponenten physikalisch in das Zielsystem kopiert werden. Er unterscheidet sich damit

signifikant von einer aktiven Code-Generierung. Ein großer Vorteil dieses Konzeptes ist, dass hierbei der Aufwand für die Verifikation entfällt, der bei einer aktiven Code-Generierung je nach Auslegung einmalig oder wiederholt anfällt. Weiterhin vorteilhaft ist, dass auch der d3web-Kern, also die Ausführungseinheit, bei jedem Transfer mitgeführt wird und nicht etwa bereits im Zielsystem vorgehalten werden muss. Dadurch ist sichergestellt, dass die Versionen der Ausführungseinheit sowohl im XDK als auch im Zielsystem stets identisch sind. Technisch umgesetzt ist der Transfer durch die Make-Umgebung des Entwicklungsprojektes, die aus einem Verbund verschiedener XML-Makefiles besteht, die die einzelnen Teilaufgaben des Transfers implementieren.

Die Transferkomponenten sowie deren Formate zeigt Tabelle 5.4.

Transferkomponente	**Format**	**Bemerkungen**
Wissensbasen	XML	inklusive Media-Inhalte
d3web-Kern	Java-Bytecode	Ausführungseinheit mit Erweiterungen DiaFlux, Time-DB
Systemkonfiguration	Properties	Systemumgebung, Pfade
Resource-Bundles	Java-Bytecode	Internationalisierung gemäß I18N
Host-Konfiguration	XML, Properties	Anbindung Zielsystem
Zusatzbibliotheken	Java-Bytecode	proprietäre und Drittanbieter-Tools

Tabelle 5.4 – *Transferkomponenten für ein Zielsystem*

Anders als bei einer konventionellen Software-Entwicklung, die in der Regel aus den Phasen Editieren-Kompilieren-Ausführen-Testen-Ausliefern besteht, erweitert sich der Lebenszyklus von mit dem Software-Framework entwickelten wissensbasierten Software-Anwendungen wie in Abbildung 5.5 gezeigt. Zusätzlich zu obigen Phasen schließen sich nach der Modellierung der Transfer ins Zielsystem sowie der Durchlauf der Phasen Kompilieren-Validieren-Ausliefern an.

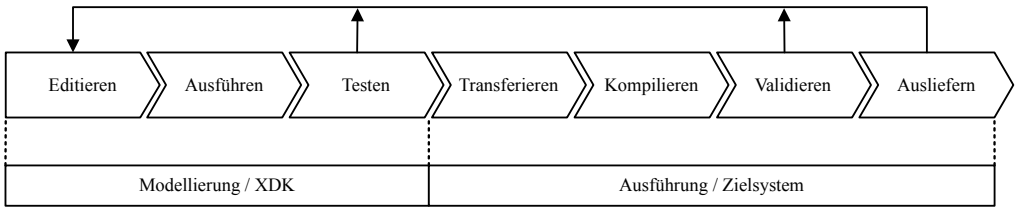

Abbildung 5.5 – *Lebenszyklus im Software-Framework*

5.4 AUSFÜHRUNG UND ANWENDUNGSSTEUERUNG

Entlang der Entstehung einer wissensbasierte Software-Anwendung sind wir nun bei der Ausführung eben solcher in einem oder mehreren Zielsystemen angelangt. Zielsysteme können alle Hardware- und Software-Plattformen sein, die über ein Java-Laufzeitsystem verfügen und bidirektionalen Zugriff auf ihre Datenbereiche ermöglichen. Dies können sowohl konventionelle, solitäre Computer-Systeme mit Betriebssystemen wie Windows, MacOs oder Linux sein als auch Mikrokontroller-gesteuerte, eingebettete Systeme mit Echtzeitbetriebssystemen wie Windows CE, vxWorks, QNX, LynxOS oder proprietäre Mikrokernel, wenn sie denn eine Java-Umgebung unterstützen. Bei medizintechnischen Systemen wie Beatmungsgeräten, Patientenmonitoren, Narkosesystemen etc. handelt es sich überwiegend um eingebettete Echtzeitsysteme.

Mit einem Datenmodell und der Anwendungssteuerung wird die in Abschnitt 4.3 vorgestellte Systemarchitektur quasi „zum Leben erweckt" und die in Abschnitt 4.2 spezifizierten Anwendungsfälle implementiert.

Repository. Das Repository ist das intrinsische Datenmodell des Software-Frameworks und der zentrale, persistente Speicherort für alle Datenobjekte. Es ist als Java-Klassenhierarchie bestehend aus einer Menge von abstrakten Klassen ausgelegt, um auch hier den Anteil invarianter Komponenten so hoch wie möglich zu halten. Praktisch bedeutet dies, dass für jede neue Wissensbasis die abstrakten Objekte spezifisch für den jeweiligen klinisch-therapeutischen Prozess zu implementieren sind. Das Repository kapselt die globalen, fachlichen Nutzdaten der wissensbasierten Software-Anwendung, die von den Subsystemen produziert, konsumiert und zwischen ihnen ausgetauscht werden. Lokale Daten dürfen nur innerhalb eines Subsystems existieren und bleiben den restlichen Subsystemen verborgen.

Abbildung 5.6 zeigt auszugsweise das Datenmodell des Repository.

Es gibt die generische Oberklasse *ApplicationObject*, von der alle weiteren Entitäten abgeleitet sind. Für die Entität *Patient* ist beispielhaft eine konkrete Ableitung angegeben. Mit dem Repository und seiner Klassenhierarchie ist folglich die fachliche Ontologie einer wissensbasierten Software-Anwendung bestimmt.

Ereignisse. Eine Menge vordefinierter Ereignisse (engl. Events) sorgen für die Dynamik des Gesamtsystems während der Ausführung. Analog zum Repository sind sie als abstrakte Klassen ausgelegt, die für den jeweiligen klinisch-therapeutischen Prozess auszuformulieren sind und die von der generischen Oberklasse *ApplicationEvent* abstammen. Die strikte Autonomie der einzelnen Subsysteme ist eine entscheidende Voraussetzung, um Erweiterbarkeit, Portierbarkeit, Wartbarkeit und Wiederverwendung des Software-Frameworks zu gewährleisten.

Abweichend von der klassischen Ereignissteuerung, bei der Events grundsätzlich parameterlos sind, können Ereignisse im Software-Framework parametrisiert werden. Und zwar ausschließlich mit Objekten aus dem Repository. Auf diese Weise lösen Events nicht nur asynchrone Signale aus, sondern transportieren zusätzlich Nutzdaten zwischen den Subsystemen. Mit dieser Zusatzfunktion entsteht eine weitere Kapselung innerhalb des Systems. Die Menge der Events

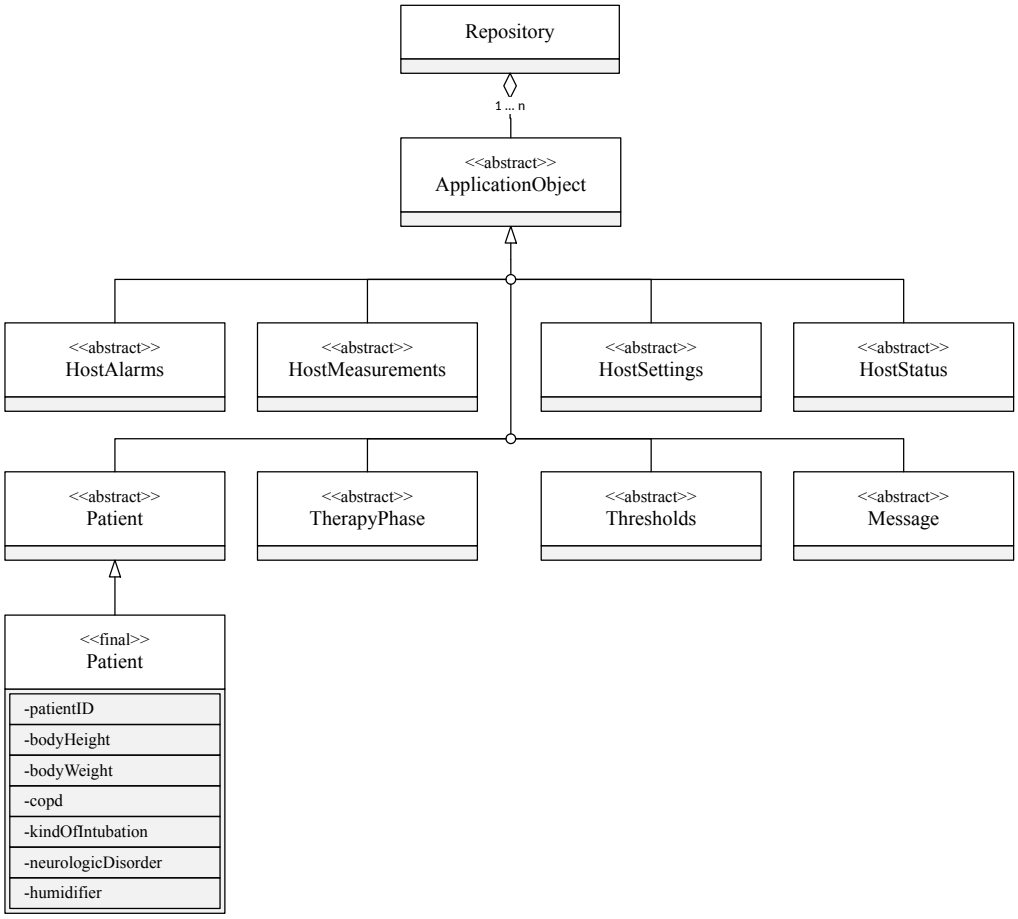

Abbildung 5.6 – *Ontologie des Repository*

im Software-Framework ist fest vorgegeben, invariant und programmatisch in der globalen *Eventlist* niedergelegt.

Die Anwendungsfälle mit ihren Szenarien und Interaktionen bestimmen zusammen mit technischen Abläufen das Laufzeitverhalten des Gesamtsystems. Neben dem Transport von Ereignissen innerhalb der Subsysteme übernehmen die Anwendungssteuerung und die Subsysteme der Vermittlungsschicht die essenzielle Aufgabe der Ereignissteuerung. Für jedes einzelne Event ist ein Kontext, Quell- und Zielsubsystem sowie zu transportierende Datenobjekte festgelegt, sodass sich eine fixe Reihenfolge von Ereignisbewegungen ergibt. Wird zu irgendeinem Zeitpunkt während der Ausführung diese Reihenfolge nicht eingehalten, so führt dies zu einem fatalen Systemfehler und zum Abbruch der gesamten Anwendung.

Während das Subsystem *Anwendungssteuerung* alle Ereignisse der Eventlist überwacht, prüfen die einzelnen *Vermittler* der Anwendungsschicht nur die für sie relevanten Events. Tabelle 5.5 zeigt von den derzeit 28 Events die wichtigsten. In der Implementierung wird das Subsystem *Anwendungssteuerung* als *MasterControl* bezeichnet. Viele der Ereignisse treten paarweise auf, um die Korrektheit einer Transaktion systemisch sicherstellen zu können. Zu erkennen sind diese Paare an

den Namensendungen *Requested/Confirmed, Requested/Done* oder *Started/Closed*. Das Ereignis *informationRequested* nimmt eine Sonderstellung ein, denn mit ihm lassen sich zu beliebigen Zeitpunkten beliebige Repository-Objekte von und zu beliebigen Subsystemen transportieren, was zum Beispiel für den Anwendungsfall *Zeige Falldaten an* benötigt wird. Fachliche Situationen, die zu einem Abbruch einer laufenden Patientensitzung, eines Falles, via *abortOfPatientSessionRequested* führen, ergeben sich aus dem Domänenwissen. Sie beschreiben Szenarien, die das System aus seinen Spezifikationen, also der Zweckbestimmung, führen, wie beispielsweise eine nicht tolerierbare Verschlechterung des Patientenzustandes.

Ereignis	**Semantik**	**Quelle**	**Ziel**
startOfApplicationSessionRequested	WBSA soll gestartet werden	MasterControl	XPS HOST GUI
startOfPatientSessionRequested	Patientensitzung (Fall) soll gestarted werden	GUI	XPS HOST
decisionMakingRequested	Inferenz auslösen	HOST	XPS
decisionMakingDone	Inferenz abgeschlossen	XPS	GUI HOST
changeOfDomainRequested	Wechsel der WB	GUI	MasterControl
timerElapsed	Zeitgeber ist abgelaufen	MasterControl	XPS GUI HOST
informationRequested	Repository-Objekt angefragt	XPS GUI HOST	XPS GUI HOST
abortOfPatientSessionRequested	Fachlicher Abbruch einer Patientensitzung (Fall)	XPS GUI HOST	XPS GUI HOST

TABELLE 5.5 – *Wichtige Ereignisse im Software-Framework*

Zeitgeber. Die Dynamik des Systems muss nicht nur asynchron, ereignisgesteuert ausgelegt sein, sondern auch zeitgesteuert. So lassen sich beliebige Zeitgeber (engl. Timer) definieren, die periodisch synchrone Ereignisse auslösen. Die beiden inhärenten System-Timer für jedwede WBSA sind:

1. **HostDataAcquisition** - Einlesen aller von der jeweiligen Wissensbasis erwarteten Eingangswerte vom Zielsystem. Typische Intervalle sind 2 s, 5 s, 10 s, 60 s.
2. **DecisionMaking** - Auslösen einer Inferenz (Reasoning). Typische Intervalle sind 2 min, 5 min, 10 min, 60 min.

Bestimmt durch die Frequenz des *HostDataAcquisition*-Zeitgebers sammeln sich zwischen jedem *DecisionMaking* eine Folge von Werten für alle Eingangsparameter an. In die Wissensbasis gelangt jedoch meist ein vorverarbeiteter Wert, sodass also ein retrospektiver Trend des jeweiligen Eingangsparameters zur Anwendung kommt. Es kann je Parameter jede beliebige, mathematische Funktion verwendet werden. Für die mit dem Software-Framework entwickelten wissensbasierten Software-Anwendungen kamen die Vorverarbeitungen Durchschnitt, Median,

Tief- und Hochpassfilter sowie Best-of-10 % zum Einsatz - auch kombiniert für einzelne Parameter.

Multiple Wissensbasen. Die mandatorische Systemanforderung, in der Ausführung mehrere Wissensbasen zu betreiben und zwischen diesen umzuschalten, ist unter Verwendung des Entwurfsmusters *Abstrakte Fabrik* umgesetzt und mit der Implementierung des Interface *ProblemSolver* des Subsystems XPS als Methode *DomainFactory* programmiert. Der Wechsel zwischen verfügbaren Wissensbasen kann entweder zur Kompilierzeit statisch erfolgen oder über das GUI des Zielsystems zur Laufzeit.

Multithreading. Nebenläufigkeit, asynchrones und pseudo-paralleles Verhalten und insbesondere die kontinuierliche Zusicherung der Konsistenz von Daten sind über das Thread-Modell abgesichert. Grundsätzlich ist jedem Subsystem ein eigener Thread zugeordnet, mit Ausnahme der Anwendungssteuerung, die zusätzlich noch einen weiteren Thread für jeden Zeitgeber betreibt. Eine weitere Maßnahme für die Zusicherung eines stabilen Laufzeitverhaltens ist die Programmierung der Anwendungssteuerung sowie sämtlicher Vermittler nach dem Entwurfsmuster *Singleton*, also

```
private static MasterControl masterControl = null ;
```

5.5 ANBINDUNG AN ZIELSYSTEME

Explorative Tests auf Funktionalität und Korrektheit einer erstmalig erstellten Wissensbasis sind bereits innerhalb von KnowWE möglich, ohne dass ein Transfer in ein Zielsystem stattfinden muss. Dazu durchläuft der Benutzer ein dialogbasiertes Interview (Abbildung 5.2, Seite 61), während dem er manuell alle erforderlichen Eingaben tätig und damit in der Ausführungseinheit eine Reihe von Inferenzen auslöst, die sukzessive zu einer Problemlösung führen. Ist dann der Transfer in das jeweilige Zielsystem erfolgt, vollzieht sich das Interview dagegen gänzlich dialoglos, ohne Beteiligung des Benutzers. Stattdessen liefert das Zielsystem aus seinem Datenbereich alle notwendigen Eingaben, zu denen bei Bedarf auch weitere Benutzereingaben dazu kommen können. Das GUI im Zielsystem erweitert sich entsprechend um Bereiche für die WBSA, über die während der Laufzeit die Anzeige von Meldungen, Daten, Ergebnisse, Stati, Bestätigungen etc. erfolgt.

Zwei fundamentale Schnittstellen realisieren das Paradigma der zwei Freiheitsgrade (Abschnitt 4.4) sowie die bidirektionale Interaktion mit dem Zielsystem. Sie sind hinreichend und notwendig für den Betrieb einer wissensbasierten Software-Anwendung.

1. Lesen, Verändern und Schreiben von Daten des Zielsystems über das Subsystem.
2. Einbettung der Benutzeroberfläche der wissensbasierten Software-Anwendung in das bestehende GUI des Zielsystems über das Subsystem GUI.

Augmentierung. Die konkrete, datentechnische Anbindung an ein Zielsystem geschieht über die sogenannte *Augmentierung* der Wissensbasis, die innerhalb von

KnowWE als Markup im XML-Format mit entsprechendem XSLT (Extensible Stylesheet Language Transformations) erfolgt. XSLT ist eine Sprache zur Transformation von XML-Dokumenten in andere Formate.. In der Augmentierung ist für jedes Objekt des Repository die eindeutige Zuordnung zu einem Objekt des Zielsystems, das für den Transport zuständige Event sowie weitere Konfigurationen angegeben, die u. a. das Laufzeitverhalten bestimmen. Obendrein umfasst die Augmentierung auch die Definition von Systemmeldungen, Zeitgebern und allgemeinen Systeminformationen.

Algorithmus 5.2 zeigt beispielhaft und auszugsweise eine konkrete Augmentierung.

In Zeile 2 ist der Name der Wissensbasis *CG AVent* angegeben und der Zeitgeber für Inferenzen (DecisionMaking) auf 15 s (15 000 ms) festgelegt. Im Repository selbst (ab Zeile 3 bis Zeile 32) bestimmt ein weitere Zeitgeber die Taktrate für Datenerhebungen vom Zielsystem (HostDataAcquisition) mit 4 s (4 000 ms). Beginnend mit den Zeilen 6, 12, 21 und 27 sind typische Repository-Objekte mit ihrem Kontext konfiguriert.

Erstellung und Pflege der Augmentierung ist Aufgabe des Software-Entwicklers. Sie gestaltet sich in der Praxis mühsam, ist fehleranfällig und damit zeitintensiv. Mit der Abschlussarbeit von Baier entstand ein GUI-basierter Augmentierungseditor [41], der diese Tätigkeiten deutlich vereinfacht, komfortabler gestaltet und Fehleingaben vermeidet.

```
1
2  <domain initialClassificationInterval="15000">CG Avent</domain>
3  <repository>
4      <timer-trigger id="AcquirationPeriod" value="4000"/>
5
6      <host-object id="Measure.CheckedRR" name="Checked RR" class
           ="ParameterValue">
7          <kb-object id="RR_Checked"/>
8          <export type="event" refid="DecisionMakingDoneEvent"/>
9          <adjust name="S17Setting.frequency" remote="host"/>
10     </host-object>
11
12     <host-object id="Setting.TherapyMessages" name="
           TherapyMessages" class="SettingValue">
13         <kb-object id="messages_Rated" class="messages">
14             <messages>
15                 <message id="0" class="alarm">messages_Rated.0.
                       text</message><!--
                       Adequate_respiratory_drive -->
16                 <message id="1" class="alarm">messages_Rated.1.
                       text</message><!-- Consider_recovery -->
17             </messages>
18         </kb-object>
19     </host-object>
20
21     <host-object id="Measure.RRSpn" name="SC RRspn" class="
           ParameterValue">
22         <kb-object id="RR_spon"/>
23         <derivedFrom refid="Measure.fspon"/>
24         <export type="event" refid="DecisionMakingDoneEvent"/>
25     </host-object>
26
27     <host-object id="Setting.TheGoal" name="Therapy Goal" class
           ="SettingValue">
28         <kb-object id="therapy_goal"/>
29         <import type="event" refid="
               StartOfPatientSessionRequestedEvent"/>
30         <import type="onchange"/>
31     </host-object>
32 </repository>
```

Algorithmus 5.2 – *Beispiel für die Augmentierung einer Wissensbasis*

Autonome Steuerung. Zum Schreiben von Daten des Zielsystems über das Subsystem HOST gehört ebenfalls die autonome Steuerung von Einstellparametern, die sich aus einer Episode von Inferenzen ergeben haben und den quantitativen Teil der hergeleiteten Problemlösung darstellen. Sie entsprechen der therapeutischen Intervention des Mediziners, der während einer Patientenvisite neue Einstellungen an den Medizingeräten vornimmt. Technisch gesehen bedeutet dies, dass die Software des Zielsystems für die Verstellung ihrer Einstellwerte einen weiteren datentechnischen Zugang bieten muss. Erst damit ist eine autonome Steuerung des Zielsystems durch eine WBSA möglich. Für den Fall, dass die wissensbasierte Software-Anwendung physikalisch nicht im Medizinprodukt integriert ist, sondern extern verbunden (LAN, WLAN, Bluetooth, RS232 etc.) muss noch der bidirektionale Datentransport zum Zielsystem und dort zum Modul der Einstellungen

implementiert werden.

Kann ein Zielsystem lesenden und schreibenden Zugriff auf seine Datenobjekte bieten, so eignet es sich als Plattform für die Ausführung aller denkbaren, computerisierten klinisch-therapeutischen Prozesse und ist damit zukunftssicher. Wir nennen diese Fähigkeit Read-All-Write-All (RAWA). In der Praxis werden allerdings häufig zunächst nur die für eine bestimmte wissensbasierte Software-Anwendung benötigten Parameter zugreifbar gemacht, was einem Read-All-Write-Some (RAWS) entspricht.

Betriebsarten. Eine zentrale Anforderung des SWFW ist die Bereitstellung von drei Betriebsarten, mit denen eine wissensbasierte Software-Anwendung ausgeführt werden kann, siehe auch Abschnitt 1.4. Diese Betriebsarten definieren, wie Problemlösungsergebnisse der Wissensbasis zyklisch an das Zielsystem und den Benutzer weiter zu geben sind.

Am wenigsten invasiv ist der Open-Loop-Modus, auch als Entscheidungsunterstützung bezeichnet, bei dem neu ermittelte Einstellwerte nicht zu einer autonomen Verstellung führen, sondern dem Benutzer lediglich via GUI angezeigt werden. Einen Schritt weiter geht der Modus der Halbautomatik (Semi-Closed-Loop), der seine Problemlösungsergebnisse dem Benutzer präsentiert, jedoch erst nach Bestätigung durch diesen die autonome Verstellung vornimmt. Der Funktion eines Autopiloten gleich ist schließlich die Vollautomatik (Closed-Loop), die vollständig ohne Interaktion mit dem Benutzer auskommt und die autonome Verstellung unmittelbar ausführt. Die jeweils gewünschte Betriebsart ist alleinig zur Kompilierzeit festlegbar und technisch realisiert, indem eine Mehrkanalität zu den relevanten Parameter geschaffen wurde, mit der der Datenstrom entweder nur zum Benutzer, nur zum Zielsystem oder von und zu beiden geregelt wird.

Entwicklungsprojekt. Die initialen Rahmenbedingungen für die Entwicklung einer wissensbasierten Software-Entwicklung spielen eine ganz erhebliche Rolle bezüglich zeitlichem und monetären Aufwand, Projektrisiken, technischer Machbarkeit und Funktionsumfang.

Ideale Bedingungen finden sich, wenn Zielsystem und wissensbasierte Software-Anwendung gemeinsam und in Java neu zu entwickeln sind. Hier kann von Anbeginn auf alle technischen Gegebenheiten adäquat eingegangen werden. Komplexer ist dagegen die Neuentwicklung eines nicht Java-basierten Zielsystems und einer zu integrierenden wissensbasierte Software-Anwendung. Zwar lässt sich eine adäquate Java Virtual Machine (JVM) für die Hardware des Zielsystems noch relativ leicht beschaffen, doch ist die technische Verbindung zwischen den „beiden Welten" anspruchsvoll. Weit verbreitet und auch für diese Arbeit berücksichtigt sind Programmiersprachen wie C und C++ für die Nicht-Java Umgebung. Die Kooperation zwischen diesen beiden geschieht mit dem Java Native Interface (JNI) [43].

Die größte Herausforderung liegt jedoch in der Entwicklung einer wissensbasierten Software-Anwendung für ein bereits bestehendes, nicht Java-basiertes Zielsystem. Neben der Aufgabe, beide Systeme miteinander zu koppeln - ebenfalls via JNI - stellen sich essenzielle Fragen wie: Gibt es für diese Konstellation eine adäquate JVM? Lassen sich alle notwendigen Parameter des Zielsystems autonom steuern?

Reichen die Betriebsmittel, also CPU-Leistung, Speicher und andere? Die Antworten auf derlei Fragen sind mühsam zu finden, entscheiden jedoch über Erfolg und Kosten des Entwicklungsprojektes.

Abschließend sei ein Beispiel zur Verdeutlichung der technischen Verbindung zwischen Java und C/C++ gegeben, siehe Algorithmus 5.3. Das Exzerpt zeigt die Initialisierung der autonomen Steuerung (ExternalControl) eines Zielsystems, hier: das Beatmungsgerät Evita XL.

Implementiert ist die Funktion `generateExternalControlId` als erster Schritt der Initialisierung. Im C++-Modul `ExternalControl.h` (Zeilen 2 bis 13) ist die Signatur der Funktion, erweitert um JNI-Spezifika, definiert. Im C++-Modul `ExternalControl.cpp` (Zeilen 16 bis 24) wird die Funktion gemäß dieser Signatur implementiert, um schließlich im Modul `ExternalControl.java` (Zeilen 27 bis 41) in die Java-Laufzeitumgebung zu gelangen.

Auch in diesem Kapitel sind sämtliche Ergebnisse zur Technologie und Implementierung durchgängig auf Konformität mit den drei Paradigmen überprüft worden, was quasi einer ersten Design-Validierung gleichkommt.

Die detaillierten Ausführungen zur Entwicklungsumgebung KnowWE mit der grafischen Notation DiaFlux verdeutlichen, wie die kollaborative Erstellung einer Wissensbasis in der Praxis von statten geht.

Die Darlegung des Transfers in das ausgewählte Zielsystem markiert eine klare Abgrenzung des Bereiches der Modellierung vom Bereich der Ausführung.

Mit dem geschilderten Laufzeitverhalten wird deutlich, wie sich das System während der Ausführung unabhängig vom jeweiligen klinisch-therapeutischen Prozess verhält.

Da die Anbindung an Zielsysteme ebenfalls beschrieben wurde, kann der Software-Entwickler neue und/oder weitere Medizinprodukte technisch mit einer wissensbasierten Software-Anwendung verbinden.

```

/* ----- ExternalControl.h ----- */

#include <jni.h>

/*
 * Class:     s420_env_wrapper_EC
 * Method:    generateExternalControllerId
 * Signature: (IBZ)I
 * Java:      generateExternalControllerId( COMP_ID, SUBCOMP_ID
     , true )
 */
JNIEXPORT jint JNICALL
    Java_s420_env_wrapper_EC_generateExternalControllerId
  (JNIEnv *, jobject, jint, jbyte, jboolean);


/* ----- ExternalControl.cpp ----- */

#include "ExternalControl.h"

JNIEXPORT jint JNICALL
    Java_s420_env_wrapper_EC_generateExternalControllerId
  (JNIEnv *, jobject, jint aComponentKey, jbyte aSubcomponent,
      jboolean requestStart)
{
    return S17::S17ExternalControlConfiguration::
        generateExternalControllerId( aComponentKey,
        aSubcomponent, requestStart ) ;
}


/* ----- ExternalControl.java ----- */

static {

    System.loadLibrary( "S42_WrapperEC" );
    TraceConsole.println( MasterControl.SUBSYS_ID_ENV, "
        S42_WrapperEC.dll loaded!" );
}

/*
 * API for using External Control to HOST objects
 */
public final void startExternalControl() {

    generateExternalControllerId( COMP_ID, SUBCOMP_ID, true );
}
```

ALGORITHMUS 5.3 – *Beispiel für eine JNI-Transaktion*

KAPITEL 6

EVALUIERUNG

An diesem Punkt ist die Entwicklung einer oder mehrerer wissensbasierter Software-Anwendungen mit dem Software-Framework fertig gestellt worden. Nun schließt sich die essenzielle Aufgabe der systematischen Evaluierung der erreichten Ergebnisse an.

Einleitend geben wir den normativen Rahmen für diese komplexen und anspruchsvollen Tätigkeiten vor. Es ist wichtig, den Umfang der Evaluierung zu klären und zwar sowohl für das Software-Framework selbst, als auch für damit generierte Software-Anwendungen.

Bevor wir auf die einzelnen Verfahren eingehen, definieren wir die Begriffe Verifizierung und Validierung als Voraussetzung für eine erfolgreiche Evaluierung.

Es folgt die detaillierte Schilderung derjenigen Evaluierungsverfahren, die zu diesem Zwecke ausgewählt wurden. Dies sind die in silico Simulation per Software, die Simulation mittels einer patientenähnlichen Hardware sowie die Durchführung klinischer Studien.

Schlussendlich muss eine wie zuvor beschrieben entwickelte und qualitätsgesicherte Software-Anwendung in den dafür vorgesehenen Ländern zugelassen werden. Dazu gehören auch Folgeaktivitäten wie die klinische Nachbeobachtung. Die Beschreibung der dafür notwendigen Aktivitäten und Herausforderungen schließen dieses Kapitel.

6.1 NORMEN UND RICHTLINIEN

Neben der unternehmerischen Selbstverständlichkeit, Produkte in höchster Qualität zu entwickeln und in den Markt zu bringen - auch Good Engineering Practice (GEP) genannt - existiert ein breites Normenwerk, das dies auch legislatorisch vorschreibt und von entsprechenden Institutionen regelmäßig formal überprüfen lässt. Das betrifft insbesondere stark regulierte Branchen wie die Medizintechnik. Nachfolgend sind diejenigen internationalen Normen aufgeführt, die für medizinische Software im Allgemeinen und für die Themen dieser Arbeit im Besonderen relevant sind. Dabei ist zu unterscheiden zwischen der Evaluierung des Software-Frameworks als generatives System und den damit erstellten wissensbasierten Software-Anwendungen (WBSA).

ISO 9001. *Qualitätsmanagementsysteme - Anforderungen*
Die ISO 9001 ist eine international anerkannte Norm, die Anforderungen an ein Qualitätsmanagementsystem (QMS) für Unternehmen und Organisationen jeder Größe und Branche definiert. Sie ist Teil der ISO 9000-Familie und gilt als Grundlage für die Implementierung eines effektiven Qualitätsmanagements.

ISO 13485. *Medizinprodukte - Qualitätsmanagementsysteme - Anforderungen für regulatorische Zwecke*
Die ISO 13485 definiert spezifische Anforderungen an ein Qualitätsmanagementsystem für die Hersteller von Medizintechnik und ist damit eine Spezialisierung der ISO 9001 für eben diese Branche. Sie wird weltweit von Unternehmen, die Medizinprodukte entwickeln, herstellen, vertreiben und warten, angewendet, um die Produktqualität und die regulatorischen Anforderungen zufriedenstellend zu erfüllen.

ISO 14971. *Anwendung des Risikomanagements auf Medizinprodukte*
Die ISO 14971 legt Anforderungen für das Risikomanagement von Medizinprodukten fest. Diese Norm ist speziell darauf ausgerichtet, die Sicherheit von Medizinprodukten während ihres gesamten Lebenszyklus zu gewährleisten, indem sie Vorgaben für Identifikation und Mitigation von Produktrisiken macht.

IEC 62304. *Medizinprodukte-Software – Software-Lebenszyklus-Prozesse*
Die IEC 62304 ist eine internationale Norm, die spezifische Anforderungen an den Lebenszyklus von Software für Medizinprodukte bestimmt. Sie bietet einen strukturierten Rahmen für die Entwicklung und Wartung von Software, die in Medizinprodukten verwendet wird, und stellt sicher, dass diese sicher und zuverlässig ist.

IEC 62366. *Anwendung der Gebrauchstauglichkeit auf Medizinprodukte*
Die IEC 62366 gibt Empfehlungen zur Anwendung von Gebrauchstauglichkeit (engl. Usability) für Medizinprodukte vor. Ihr Ziel ist es, sicherzustellen, dass Medizinprodukte effektiv, sicher und benutzerfreundlich gestaltet werden, um das Risiko etwaiger Benutzerfehler zu minimieren.

DIN EN ISO 14155. *Klinische Prüfung von Medizinprodukten an Menschen - Gute klinische Praxis*
Die DIN 14155 regelt die Durchführung von klinischen Prüfungen mit Medizinprodukten. Sie bietet einen internationalen Standard, der ethische und wissen-

schaftliche Anforderungen an die Planung, Durchführung, Überwachung und Dokumentation solcher Evaluierungen beschreibt. Sie ist entscheidend für die Konformität, Glaubwürdigkeit und weltweite Akzeptanz der Ergebnisse klinischer Studien. Außerdem stellt sie sicher, dass Entwicklung und Bewertung von Medizinprodukten gemäß den höchsten ethischen und wissenschaftlichen Standards erfolgen.

Typischerweise ist die konkrete Umsetzung aller Normenvorgaben innerhalb der Unternehmen durch eine Menge von Geschäftsprozessen, interner Standards, Arbeitsanweisungen, Konventionen, IT-Werkzeugen etc. geregelt, die zusammen das Qualitätsmanagementsystem (QMS) der Organisation ausmachen. Das QMS wiederum ist zusammen mit weiteren Managementsystemen wie Umweltmanagement (ISO 14001), Gesundheits- und Sicherheitsmanagement (ISO 45001), Informationssicherheit (ISO 27000) in das integrierte Managementsystem (IMS) zu einem eigenen Framework gebündelt.

6.2 UMFANG DER EVALUIERUNG

Aufwand und damit einhergehend Planbarkeit, Risiko und Dokumentation einer vollständigen Evaluierung kann gerade bei stak regulierten Systemen beträchtlich sein. Jedoch wird mit dem generativen Ansatz des Software-Frameworks plus der strikten Trennung invarianter von variablen Komponenten der Gesamtaufwand nicht nur geringer, sondern auch reproduzierbar und bestenfalls auch automatisierbar.

Evaluierung des Software-Frameworks. In den vorangehenden Kapiteln haben wir, beginnend bei Konzeptionierung über Anforderungsmanagement, Entwurf und Systemarchitektur bis hin zur Implementierung, durchgängig invariante und variable Anteile identifiziert und abgegrenzt. Für die Evaluierung des Software-Frameworks ergibt sich damit der Vorteil, dass die invarianten Bestandteile nur einmalig zu prüfen und freizugeben sind. Sie bleiben danach solange von weiteren Evaluierungen unberührt, bis funktionale Änderungen am Software-Framework von Nöten sind. Auch für nicht selbst entwickelte, mitverwendete Systemkomponenten, wie d3web und KnowWE, findet dieses Evaluierungsprinzip Anwendung, denn jene zählen ebenfalls zu invarianten Bausteinen.

Die zentralen, inhaltlichen Anforderungen für die Evaluierung des Software-Frameworks sind mit der Zweckbestimmung und den Anwendungsfällen gegeben, siehe Abschnitt 4.2. In der Systemarchitektur sind es die grau hinterlegten Komponenten, wie in Abbildung 4.2, Seite 46 illustriert.

Evaluierung der wissensbasierten Software-Anwendung. Eine mit dem Software-Framework entwickelte wissensbasierte Software-Anwendung (WBSA) computerisiert einen klinisch-therapeutischen Prozess (KTP) via Modellierung in eine Wissensbasis und stellt diese für die Ausführung in einem Zielsystem bereit. Grundforderung ist eine hohe Flexibilität und Reaktionszeit bezüglich Änderungen an diesen domänenspezifischen, variablen Inhalten.

Mit dem Paradigma des zweiphasigen Nutzungszyklus (Abschnitt 4.5) ist dies gegeben. Gemäß des universellen Prinzips der Single Source of Truth umfasst

die Wissensbasis auch Zweckbestimmung und Lastenheft des jeweiligen klinisch-therapeutischen Prozesses. Zu evaluieren sind daher ausschließlich die Wissensbasis selbst sowie die variablen Anteile des GUI und der Zielsystemanbindung. Per Definition sind im Betrieb der WBSA Änderungswünsche am klinisch-therapeutischen Prozess deutlich häufiger zu erwarten als an den übrigen, variablen Komponenten. Da sich die Evaluierung des klinisch-therapeutischen Prozesses ausschließlich auf die Wissensbasis beschränkt, sollte sie effizient, reproduzierbar, zuverlässig und idealerweise (teil)automatisiert ablaufen.

Die Evaluierung beginnt mit der akribischen Festlegung und Planung von Umfang, Verfahren, Ergebniserwartung, IT-Werkzeugen etc. und wird im *Verifikations- und Validierungsplan* dokumentiert. *Verifikations- und Validierungsspezifikationen*, auch Testspezifikationen genant, beschreiben detailliert, wie jede einzelne Evaluierungsmaßnahme durchzuführen ist. Schließlich liefert die dokumentierte Ausführung der Testspezifikationen durch den Testingenieur die Auswertung in Form des *Verifikations- und Validierungsreport*

Nachfolgend wenden wir uns der Durchführung der Evaluierung zu, wobei der Schwerpunkt auf der besonderen Methodik des Software-Frameworks und der Charakteristika der wissensbasierten Software-Anwendungen liegt.

6.3 VERIFIZIERUNG

Definition VERIFIZIERUNG

> Bestätigung durch Bereitstellung eines objektiven Nachweises (...), dass festgelegte Anforderungen (...) erfüllt worden sind.
> ISO 9000:2015 [112]

Diese der ISO 9000 entstammende Definition der Verifizierung als die initiale Evaluierungsaktivität bleibt recht vage und bedarf einer Konkretisierung, um sie für Software anwenden zu können:

Verifizierung sind alle Aktivitäten, mit denen überprüft wird, ob eine Software genau nach den spezifizierten Anforderungen und Entwurfsvorgaben entwickelt wurde. Ziel der Verifizierung ist es sicherzustellen, dass die Software korrekt funktioniert und alle definierten Anforderungen erfüllt.

Die Leitfrage der Verifizierung lautet also: *„Haben wir das System korrekt gebaut?"*

Ein effektives und im Bereich der medizinischen Software-Entwicklung weit verbreitetes Verfahren zur Verifikation sind die *Code-Reviews*. Eine Studie von De Silva et al. hat ergeben, dass Code-Reviews signifikant zur Reduzierung von Fehlern und zur Verbesserung der Wartbarkeit beitragen [64]. Unter Code-Reviews versteht man das Prüfen des nicht kompilierten Quelltextes durch weitere Personen beispielsweise im Rahmen von Inspektionen. Bei solchen Inspektionen stellt der Autor eines Code-Fragmentes seinen Quelltext direkt am Bildschirm ein oder mehreren Kollegen vor, die entsprechende Fragen stellen und/oder detaillierte Erläuterungen verlangen. Ziel dieses „laut Vorlesens" ist das Aufdecken etwaiger Fehler im Quelltext, was erstaunlich gut gelingt, meistens übrigens durch

den Autor selbst. Code-Reviews sind ebenso effektiv wie sie auch zeitaufwändig sind, sowohl in der Vorbereitung als auch in Durchführung und Nachbearbeitung. Es empfiehlt sich daher eine sorgfältige Planung unter Abwägung, welche Software-Komponenten überhaupt einem Code-Review unterzogen werden sollen. Minimalumfang sollten alle in der Risikoanalyse des Projektes gemäß ISO 14971 identifizierten, sicherheitskritischen Module sein. Ebenfalls empfehlenswert ist das vorherige Nutzen von Werkzeugen zur statischen Code-Analyse und Berechnung von Metriken, um möglichst viele Anomalien bereits im Vorwege zu eliminieren.

Im hier vorliegenden Fall sind initial alle Quelltexte von Software-Framework und jeweiliger wissensbasierter Software-Anwendung mit Hilfe von Code-Reviews inspiziert worden. Nach dieser Erstprüfung war dies nur noch für variable Software-Module notwendig, falls diese während des Lebenszyklus geändert werden mussten.

6.4 VALIDIERUNG

Definition VALIDIERUNG

> Bestätigung durch Bereitstellung eines objektiven Nachweises (...), dass die Anforderungen (...) für einen spezifischen beabsichtigten Gebrauch oder eine spezifische beabsichtigte Anwendung erfüllt worden sind.
> ISO 9000:2015 [112]

Wie schon bei der Verifizierung, reicht auch für die Definition der Validierung die Festlegung in der ISO 9000 allein nicht aus. Wir verstehen unter Validierung von Software einen Prozess, bei dem sichergestellt wird, dass eine Software-Anwendung tatsächlich für ihren beabsichtigten Zweck geeignet ist und die Bedürfnisse der Endnutzer erfüllt. Während die Verifizierung prüft, ob die Software gemäß den Spezifikationen korrekt implementiert wurde, geht die Validierung darüber hinaus und stellt sicher, dass die Software in der realen Anwendung ihren deklarierten Zweck erfüllt.

Leitfrage der Validierung ist also: *„Haben wird das korrekte System gebaut?"*

Konkreter und noch unabhängig von spezifischen Evaluierungsverfahren bedeutet Validierung, dass alle inhaltlichen Produkt- und Systemanforderungen aus allen vorhandenen Spezifikationsdokumenten auf Erfüllung zu überprüfen sind. Mit Spezifikationsdokumenten sind Lastenheft, Pflichtenheft, Architektur- und Software-Spezifikationen sowie alle weiteren Systemanforderungen unterschiedlicher Detailtiefe und Ausrichtung gemeint. Es sei angemerkt, dass bei jeglichen Validierungen stets auch und informell „mitverifiziert" wird, weswegen auch die Reihenfolge - erst Verifizierung, dann Validierung - plausibel und wichtig ist.

Abbildung 6.1 zeigt Evaluierungsverfahren, die wir für diese Arbeit genutzt haben sowie deren zeitliche Anordnung als auch deren Effektivität und Aufwand.

Von den Software-internen Verfahren bis hin zur klinischen Studie nehmen Aufwand, Kosten und Komplexität zwar kontinuierlich – und zum Teil beträchtlich - zu, aber eben auch Güte, Nutzen und Verlässlichkeit.

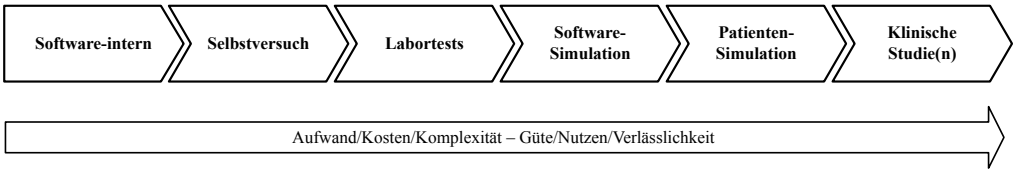

ABBILDUNG 6.1 – *Gängige Evaluierungsverfahren*

Für die Belange des Software-Frameworks und der wissensbasierten Software-Anwendungen als besonders geeignet haben sich die Testverfahren Software-Simulation, Patienten-Simulation und klinische Studie erwiesen. Ihnen ist jeweils ein eigener Abschnitt gewidmet. Die übrigen sind im Folgenden näher beschrieben.

Allen Testverfahren gemein ist, dass sie die zentrale Frage beantworten müssen, die die Natur eines klinisch-therapeutischen Prozesses mit sich bringt:

Wie können wir das physiologische Verhalten eines Patienten nachbilden?

Software-intern. Hierunter fallen sämtliche Bemühungen, bereits im Quelltext der Software Testverfahren zu hinterlegen, die dann während der Laufzeit zur Anwendung kommen und die dabei entstehenden Ergebnisse auch gleich dokumentieren. Gerade im Zuge der hochdynamischen, agilen Software-Entwicklung sind diese Ansätze populär geworden und haben den Test-First-Ansatz (schreibe erst den Test, dann den fachlichen Code), die Unit-Tests (jedes Modul braucht seinen eigenen Test) und schließlich das Behaviour-driven Testen (BDT) hervor gebracht. Letzteres verfolgt die Idee, den Fokus auf das Verhalten einer Anwendung zu legen und Tests aus Sicht der Benutzeranforderungen und der erwarteten Funktionalität zu formulieren. BDT kam auch im Rahmen dieser Arbeit experimentell zum Einsatz, wie von Hatko et al. 2014 beschrieben [20].

Selbstversuch. Der Entwickler wird zum Patienten.
Natürlich nur in sehr eingeschränktem und niemals die Gesundheit gefährdenden Maße. Dennoch kann es - gerade in der Anfangsphase, nach Abschluss der ersten Programmierdurchläufe - sinnvoll sein, beispielsweise einfach einmal selbst am Beatmungsgerät mit einer Maske zu atmen und die Reaktion des computerisierten klinisch-therapeutischen Prozesses ad hoc zu erfahren. Derartige Selbstversuche sind mit Bedacht zu absolvieren, rein explorativ, informell und undokumentiert. Dennoch bieten sie wertvolle erste Erkenntnisse und Einsichten zur Wirksamkeit einer wissensbasierten Software-Anwendung.

Labortests. In den Laboren der Medizintechnikhersteller findet sich in der Regel hinreichend Equipment, um in Entwicklung stehende Medizinprodukte umfänglich zu testen. Im Bereich der maschinellen Beatmung stehen dafür sogenannte Lungensimulatoren zur Verfügung, die mechanisch die Haupteigenschaften der menschlichen Lunge nachbilden und in der einfachsten Form aus einem Balg (Lungenvolumen) bestehen, einer Rückholfeder (Elastizität) und einem mechanischen Widerstand (Resistenz), gegen den der Balg befüllt und entleert werden muss. Neuere Generationen von Lungensimulatoren sind komplexer ausgestattet, gehen

mechanisch präziser auf die Physiologie ein, werden elektronisch gesteuert und können damit eine statische Spontanatmung erzeugen, mit der eine erste Rückkopplung an ein Zielsystem möglich ist, das einen zu testenden, computerisierten klinisch-therapeutischen Prozess für Beatmung ausführt.

Einige der modernen Lungensimulatoren wie beispielsweise der ASL5000® der Firma IngMar Medical erlauben eine autonome Steuerung ihrer Systemparameter von außen über Skripte, sodass sich auch umfangreichere, klinische Episoden eines spontan atmenden Patienten programmieren und wiederverwenden lassen. Auch kommen zunehmend mathematische Modelle zur Simulation physiologischer Vorgänge zum Einsatz, um noch mehr und realitätsgetreue Patientenreaktionen erzeugen zu können.

6.5 SOFTWARE-SIMULATION

Lungensimulatoren gestatten eine initiale Einbindung eines virtuellen Patienten in die Evaluierung wissensbasierter Software-Anwendungen in der Anästhesiologie. Nachteilig ist jedoch die aus der Herstellerabhängigkeit resultierende, fehlende Erweiter- und Anpassbarkeit, die hohen Anschaffungskosten sowie die Limitierung auf Beatmung und Lungenphysiologie. Deutlich vorteilhafter sind Simulationen, die gänzlich in Software realisiert sind, und gänzlich ohne Hardware und anderer Physik auskommen. Sie werden häufig auch in silico Computer-Simulationen genannt. In silico Evaluierungen tragen dazu bei, Zeit und Kosten zu sparen, Risiken zu minimieren, zulassungsrelevante Nutzungsdaten zu generieren und die Hypothesenbildung und -prüfung zu beschleunigen.

Für die vorliegende Arbeit wurde die Software-Simulation *SmartPatient* entwickelt und zur Evaluierung etlicher mit dem Software-Framework generierter wissensbasierter Software-Anwendungen eingesetzt. Grundidee des SmartPatient ist, ein Evaluierungswerkzeug zu erschaffen, das offen, flexibel und erweiterbar ist und einen unter Therapie mit einem oder mehreren Medizinprodukten stehenden Patienten und dessen physiologische Reaktionen adäquat und dynamisch nachbildet. Außerdem soll das Werkzeug fernsteuerbar sein, eine Programmierschnittstelle bereitstellen und umfangreich Daten aufzeichnen, abspeichern und zurücklesen können. SmartPatient folgt damit einem Baukastenprinzip, im Verlauf seiner Evolution entsteht eine Bibliothek von Modulen aus Patiententypen, Krankheitsbildern, Therapieformen, Medizinprodukten und Ablaufmustern, die sich ad libitum zu spezifischen, wiederverwendbaren Evaluierungsszenarien zusammenstellen lassen.

In Abbildung 6.2 sind die funktionalen Komponenten der Software-Simulation dargestellt. Auch hier übernimmt eine Applikationssteuerung die gesamte Dynamik des Systems und sorgt damit für die Erfüllung oben genannter Zweckbestimmung.

Die beiden wesentlichen Anwendungsszenarien sind: Der *Benutzer* bedient den SmartPatient über das Graphical User Interface (GUI), mit dem er Eingaben tätigen sowie Messwerte und grafische Echtzeitkurven betrachten kann. Das ist probat für explorative Untersuchungen, kurze Testsequenzen und die Überprüfung dedizierter Hypothesen. Sollen dagegen automatisierte Langzeittest durchgeführt werden, die zuweilen wochenlang laufen, so können *Software-App 1… L* die dafür

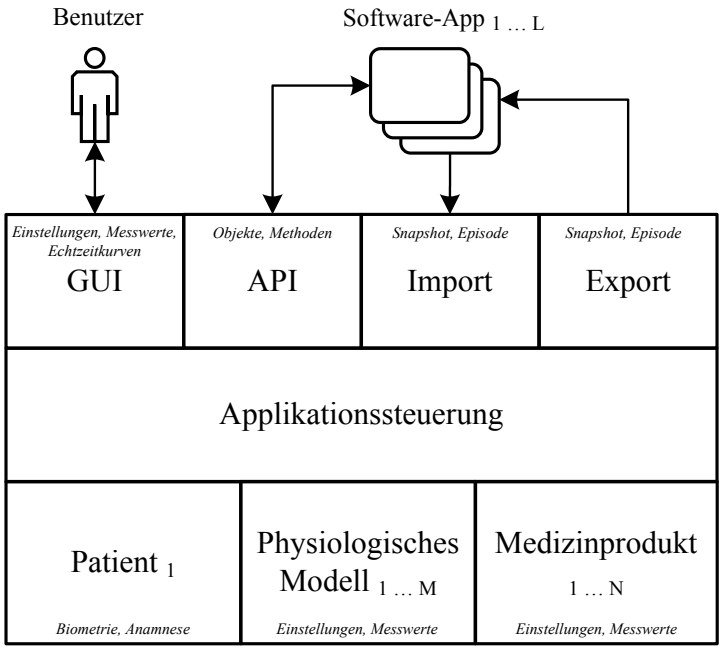

Abbildung 6.2 – *Systemkomponenten der Software-Simulation SmartPatient*

vorgesehene Programmierschnittstelle *API* (Application Programming Interface) nutzen. Die API selbst ist mittels Remote Message Invocation (RMI) in Java implementiert worden, um den bidirektionalen Zugriff auf Java-Objekte und den externalen Aufruf von Java-Methoden (Remote Procedure Call) des SmartPatient realisieren zu können.

Das Datenmodell (vgl. Repository, Abschnitt 5.4) besteht aus den Entitäten *Patient*, *Physiologisches Model 1 … M* und *Medizinprodukt 1 … N*, die jeweils einstellbar sind und spezifische Messwerte liefern. Während die inhaltliche Erweiterbarkeit durch die Adaptation der objektorientierten Repository-Entitäten erreicht wird, sorgt eine *Import-Export*-Funktionalität für die flexible Erweiterbarkeit des Laufzeitverhaltens und legt damit den Grundstein für den Aufbau einer Bibliothek von klinischen, wiederverwendbaren Szenarien für die Evaluierung. So können nicht nur alle Systemparameter des SmartPatient als sogenannter *Snapshot* für einen fixen Zeitpunkt in einem vordefinierten XML-Format abgespeichert und wieder eingelesen werden, sondern auch zeitlich unbegrenzte Sequenzen von Snapshots. Mit diesen als *Episoden* bezeichneten Aufzeichnungen ist ein Record-and-Replay-Mechanismus realisiert, der eine Langzeitevaluierung der wissensbasierten Software-Anwendung unter Verwendung klinisch realer, vom medizinischen Experten validierter Anwendungsszenarien ermöglicht.

Das erste physiologische Modell des SmartPatient diente der Simulation maschineller Beatmung und bildet das kardiorespiratorische System eines Menschen nach dem aus den 80er Jahren stammenden Standardwerk von Riley ab [49]. Die gesamte Anwendung selbst ist in Baier et al. näher beschrieben [10].

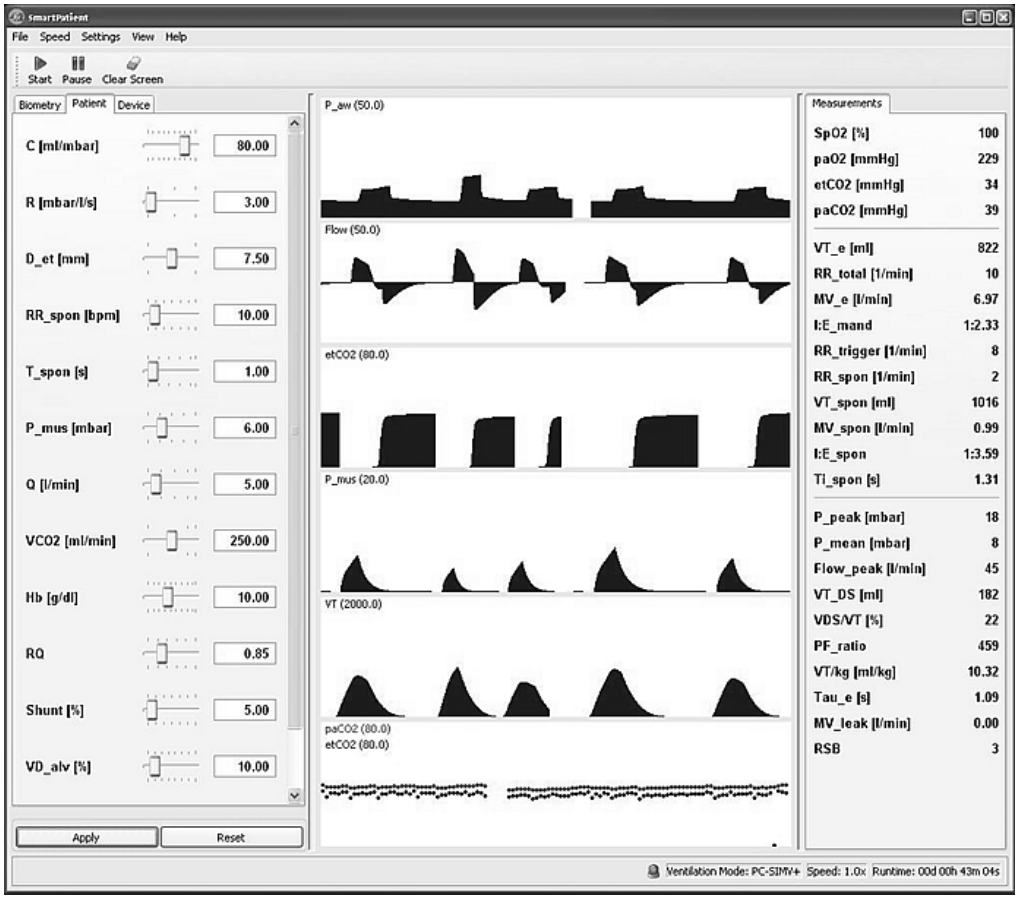

Abbildung 6.3 – *Hauptbildschirm der Software-Simulation SmartPatient*

Abbildung 6.3 vermittel einen Eindruck vom GUI des SmartPatient. Im linken, vertikalen Bereich kann der Benutzer Einstellungen zur Biometrie, zur Physiologie des Patienten und zu Einstellungen des Medizingerätes - hier: Beatmungsgerät - vornehmen. Im mittleren, vertikalen Bereich sind zentrale Messwerte grafisch in Kurvenform dargestellt. Weitere Messwerte werden im rechten, vertikalen Bereich angezeigt.

6.6 PATIENTENSIMULATION

Ein weiteres Instrument zur Beantwortung der zentralen Evaluierungsfrage, siehe Seite 82, sind die Patientensimulationen, engl. Human Patient Simulator (HPS), oft auch als Full-Scale HPS bezeichnet. Ein Full-Scale Human Patient Simulator ist eine hochentwickelte, realistisch gestaltete, mechanische und elektronische Puppe (Mannequin), die in der medizinischen Ausbildung verwendet wird, um realistische klinische Szenarien zu simulieren und deren Umgang zu schulen.

Diese Simulatoren sind so konzipiert, dass sie eine Vielzahl von anatomischen und physiologischen Reaktionen und pathophysiologischen Krankheitsbildern nachahmen können, wodurch sie ein wertvolles Werkzeug für das Training von

medizinischem Fachpersonal darstellen. Es handelt sich um äußerst komplexe Simulationssysteme, die ein nahezu vollständiges Abbild eines intensivmedizinischen Patienten liefern, einschließlich Physiologie, Pharmakologie, Pharmakokinetik, Vitaldaten-Monitoring etc.

Das Mannequin solcher HPS - zum Beispiel der US-amerikanischen Firma METI - verhält sich beinahe äquivalent zu einem humanoiden Patienten, was zu dem Begriff „True-to-Life" geführt hat. Es können Zugänge gelegt, Medikamente appliziert, mit der Puppe verbal kommuniziert und vielerlei mehr klinisch-reale Situationen geschaffen werden. Das Mannequin kann weinen, stöhnen, bluten und eine Vielzahl klinisch relevanter Reaktionen hervorrufen, die wiederum zuvor in Szenarien programmiert werden können. HPS eignen sich daher zur Durchführung von präklinischen Studien als Vorstufe zu realen klinischen Studien.

Ursprünglich für die strukturierte Aus- und Weiterbildung angehender und praktizierender Ärzte vorgesehen, eignen sich HPS auch hervorragend für die Evaluierung von wissensbasierten Software-Anwendungen. Quasi als allerletzte Evaluierungsform vor den klinischen Studien. Obwohl der Einsatz eines HPS äußerst kostspielig ist, kann er doch per Definition alle klinisch-therapeutischen Prozesse abdecken, die in seinen Einsatzbereich fallen und somit schon erste klinische Daten bezüglich Evidenz und späterer Zulassung liefern.

Im Rahmen dieser Arbeit wurde 2010 ein Human Patient Simulator eingesetzt, um die Evaluierung für einen klinisch-therapeutischen Prozess zur automatisierten, maschinellen Beatmung durchzuführen. Diese präklinische Studie hatte zum Ziel, die Sicherheit und Wirksamkeit der mit dem Software-Framework aus dem klinisch-therapeutischen Prozess entwickelten wissensbasierten Software-Anwendung (WBSA) nachzuweisen. Geleitet wurde sie durch die drei primären Fragestellungen

1. Gewährleistet die WBSA eine sichere Einstellung der Beatmungsparameter?
2. Führt die WBSA zu einer effektiven Entwöhnung von der Beatmung?
3. Reagiert die WBSA adäquat auf typische klinische und technische Zwischenfälle?

Es wurden in Summe 20 Patientenfälle (3 gesund, 17 erkrankt) und 10 Zwischenfälle für den Human Patient Simulator programmiert. Das zusammengefasste Ergebnis ist, dass die mittlere Zeitdauer bis zur Spontanatmung ($n = 8$) 18 ± 17 Minuten betrug und die Behandlungsdauer bis die Entwöhnung beendet war 122 ± 59 Minuten ($n = 3$). Sowohl Reaktion auf die Zwischenfälle als auch Reaktionszeit wurden entweder als adäquat oder unkritisch bewertet. Das Forschungsgruppe schloss daraus, dass die WBSA sicher, wirksam und stabil ist und schlugen lediglich kleinere Anpassungen vor. Die präklinische Studie und ihre Ergebnisse sind in Schädler et al. dokumentiert [2].

6.7 KLINISCHE STUDIEN

Unter dem Begriff klinische Studie lassen sich eine Vielzahl wissenschaftlicher Untersuchungsformate zusammenfassen, die das Ziel verfolgen, Sicherheit, Leistungsfähigkeit und Nutzen von Medizingeräten, Medikamenten und/oder therapeutischen Behandlungen zu testen und zu bewerten. Dazu sammeln klinische Studien, geleitet durch ein wissenschaftlich fundiertes Vorgehen *klinische Daten* als Antworten zu genau festgelegten, medizinischen Fragestellungen.

Die Medical Device Regulation (MDR) definiert klinische Daten wie folgt:

Definition KLINISCHE DATEN

> Angaben zur Sicherheit oder Leistung, die im Rahmen der Anwendung eines Produkts gewonnen werden und die aus den folgenden Quellen stammen:
> - klinische Prüfung(en) des betreffenden Produktes,
> - klinische Prüfung(en) oder sonstige in der wissenschaftlichen Fachliteratur wiedergegebene Studien über ein Produkt, dessen Gleichartigkeit mit dem betreffenden Produkt nachgewiesen werden kann,
> - in nach dem Peer-Review-Verfahren überprüfter wissenschaftlicher Fachliteratur veröffentlichte Berichte über sonstige klinische Erfahrungen entweder mit dem betreffenden Produkt oder einem Produkt, dessen Gleichartigkeit mit dem betreffenden Produkt nachgewiesen werden kann,
> - klinisch relevante Angaben aus der Überwachung nach dem Inverkehrbringen, insbesondere aus der klinischen Nachbeobachtung nach dem Inverkehrbringen.
>
> MDR, Artikel 2, Satz 48 [65]

Für eine erfolgreiche Produktzulassung ist eine *klinischen Bewertung* notwendig. Dafür liefern klinische Studien die erforderlichen klinischen Daten. Die Medical Device Regulation definiert:

Definition KLINISCHE BEWERTUNG

> Die klinische Bewertung bezeichnet einen systematischen und geplanten Prozess zur kontinuierlichen Generierung, Sammlung, Analyse und Bewertung der klinischen Daten zu einem Produkt, mit dem Sicherheit und Leistung, einschließlich des klinischen Nutzens, des Produkts bei vom Hersteller vorgesehener Verwendung überprüft wird.
> MDR, Artikel 2, Satz 44 [65]

Die *MDCG 2020-1 Guidance on Clinical Evaluation* der Medical Device Coordination Group wiederum konkretisiert die klinische Bewertung medizinischer Software in drei zu erbringende Nachweise [86]:

1. Wissenschaftliche Validität
2. Technische und analytische Leistungsfähigkeit
3. Klinische Leistungsfähigkeit

Im vorliegenden Fall bedeutet dies, das für jede mit dem Software-Framework entwickelte WBSA eine klinische Bewertung durchzuführen ist, wobei aus Projektsicht genau abzuwägen ist, welches Format der klinischen Bewertung zur Anwendung kommen soll. Klinische Studien sind generell sehr kostspielig und zeitaufwändig und stellen somit ein nicht unerhebliches Projektrisiko dar. Die DIN 14155 [108] unterstützt die Initiierung, Planung, Durchführung und Auswertung klinischer Studien im Detail.

6.8 INVERKEHRBRINGUNG VON MEDIZINPRODUKTEN

Die EU-Verordnungen stellen hohe Anforderungen an die sogenannte *Inverkehrbringung* von Medizinprodukten. In der Medical Device Regulation (MDR) findet sich dazu die folgende Definition.

Definition INVERKEHRBRINGUNG

> Die erstmalige Bereitstellung eines Produktes, mit Ausnahme von Prüfprodukten, auf dem Unionsmarkt [65]

Legislativ und regulatorisch verbindlicher Rahmen bildet die EU-Verordnung MDR 2017/745 sowie das im Mai 2011 in Kraft getretene, für Deutschland spezifisch ergänzte Medizinprodukterecht-Durchführungsgesetz (MPDG).

Eine Zulassung von Medizinprodukten gibt es im eigentlichen Sinne nicht. Stattdessen wird die Erlaubnis für den Marktzugang mittels des sogenannten *Konformitätsbewertungsverfahrens* geregelt, welches ebenfalls in der MDR beschrieben ist. In diesem erklärt ein Medizintechnikhersteller selbst das erfolgreiche Einhalten aller geltenden Bestimmungen der MDR. Art und Umfang des Konformitätsbewertungsverfahrens bestimmt die Klassifikation eines Medizinproduktes, die ebenfalls durch die MDR geregelt ist. Ist ein Produkt als Medizinprodukt qualifiziert, so kann es nach entsprechenden Kriterien in die Risikoklassen I, I*, IIa, IIb oder III eingeteilt werden. Für medizinische Software sieht die IEC 62304 ferner noch die Sicherheitsklassen A, B und C vor, welche sich auf den Umfang der zu erstellenden Dokumentation auswirken.

Regulatory Affairs. Zu deutsch: regulatorische Angelegenheiten, umfasst diejenigen Prozesse und Tätigkeiten, die sicherstellen, dass Medizinprodukte alle regulatorischen Anforderungen derjenigen Länder erfüllen, in denen sie vertrieben werden sollen. Dazu gehören vor allem die

- Einholung der erforderlichen Genehmigungen und Freigaben von den zuständigen Aufsichtsbehörden
- Einhaltung der geltenden Vorschriften und Normen

- Aufrechterhaltung der Konformität während des gesamten Produktlebenszyklus bis hin zur Außerbetriebnahme.

Zu den Aufgaben des Bereichs Regulatory Affairs zählt auch die Überwachung von Änderungen der Vorschriften und Normen sowie die Kommunikation solcher Änderungen mit den Interessengruppen innerhalb des Unternehmens, um eine kontinuierliche Einhaltung zu gewährleisten. Damit spielt Regulatory Affairs eine entscheidende Rolle, wenn es darum geht, zu gewährleisten, dass Medizinprodukte sicher und wirksam sind und den gesetzlichen Anforderungen genügen.

Klinische Nachbeobachtung. Wissen im Allgemeinen und in der Medizin, speziell der Anästhesiologie im Besonderen, ändert sich fortlaufend und oft. Für jede wissensbasierte Software-Anwendung ergibt sich damit die Notwendigkeit, ständig die Aktualität und Evidenz des genutzten Domänenwissens zu überprüfen.

Dies ist mit dem Post-Market Clinical Follow-Up (PMCF) in der MDR sogar rechtlich vorgegeben. Ein PMCF ist eine fortlaufende Überwachung und Sammlung von klinischen Daten für ein Medizinprodukt, nachdem es auf dem Markt eingeführt wurde. Der PMCF-Prozess ist ein wesentlicher Bestandteil der regulatorischen Anforderungen für Medizinprodukte und dient dazu, die langfristige Sicherheit und Leistungsfähigkeit des Produktes kontinuierlich zu bestätigen.

Zusammenfassend lässt sich feststellen, das die Evaluierung des Software-Frameworks und jeder damit entwickelten Software-Anwendung komplex und aufwändig ist und einer sorgsamen Planung bedarf. Die Auswahl der jeweiligen Evaluierungsverfahren muss sich an den normativen und gesetzlichen Vorgaben halten, aber dennoch wirtschaftlich bleiben. Auch hier kommt die Trennung invarianter von variablen Systemkomponenten nach unserem Design-Prinzip vorteilhaft zum Tragen.

Die Verwendung des SmartPatient (Abschnitt 6.5) als integrierte, umfängliche Software-Simulation für mit dem Software-Framework generierte wissensbasierte Anwendungen war - zumindest bei der Firma Dräger - neuartig und erstmalig. Gleiches gilt - in Teilen - auch für den integrativen Einsatz eines Human Patient Simulator.

KAPITEL 7

BEISPIELANWENDUNGEN

Wir haben bis hierher das Hintergrundwissen des dieser Arbeit zugrunde liegenden Themas kennengelernt, die allgemeine und spezifische Methodik des Software-Frameworks erläutert sowie die technologische Implementierung und die sich anschließende Evaluierung samt Zulassung und Inverkehrbringen vorgestellt.

In Tabelle 2.3, Seite 23 haben wir klinisch-therapeutische Prozesse aufgeführt, die sich für eine Modellierung und Ausführung als wissensbasierte Systeme für die Anästhesiologie eignen. In diesem Kapitel beschreiben wir deren Zweckbestimmung, Einsatzbereich, Zielstellung und Implementierung mit dem Software-Framework und gehen dabei auf Besonderheiten der jeweiligen Anwendung ein.

Zu diesen praktischen Beispielen zählen sowohl solche, die „reibungslos" den hier geschilderten Entstehungsprozess durchlaufen konnten und als Medizinprodukte in den Markt gebracht wurden, als auch solche, die von vorn herein für wissenschaftliche Zwecke entwickelt wurden oder aber zur Untersuchung spezieller Forschungsgebiete entstanden sind.

Ziel dieser Auflistung ist nicht nur der Nachweis der praktischen Eignung des Software-Framework, sondern auch, dessen vielseitige Einsatzmöglichkeiten und Potenziale zu verdeutlichen. Wurden für einzelne der Beispielanwendungen klinische und/oder präklinische Studien durchgeführt, so beschreiben wir diese mit den entsprechenden Referenzen.

7.1 ÜBERBLICK

Mit Entstehung und seit Verfügbarkeit des Software-Frameworks (SWFW) sind bei der Firma Dräger einige wissensbasierte Software-Anwendungen (WBSA) für die Anästhesiologie entstanden und vermarktet worden. Darüber hinaus gab es zahlreiche Forschungsansätze explorativer Art und/oder zur analytischen Untersuchung spezieller Fragestellungen. Eine Liste von Anwendungen liefert Tabelle 7.1. Diese Anwendungen werden in den folgenden Abschnitten nacheinander beschrieben und diskutiert.

Die Abkürzung CG im Projektnamen steht für „Clinical Guideline", siehe Abschnitt 2.1.4. SmartCare, Smart Sonar Sepsis und Smart Ventilation Control sind eingetragene und/oder registrierte Urheberrechte der Firma Drägerwerk AG & Co. KGaA, Lübeck.

Projektname	**klinisch-therapeutischer Prozess**	**Zielsystem(e)**
SmartCare	Entwöhnung von druckunterstützter Beatmung	Evita 4 Evita XL Evita V500 Evita V300
Smart Sonar Sepsis	Früherkennung septischer Erkrankungen	ICM (PDMS)
Smart Ventilation Control	Beatmung während Narkose	Zeus IE Atlan
Maschinelle Maskenbeatmung	Druckunterstützte Maskenbeatmung	Evita XL Evita V500
Druckkontrollierte Beatmung	Druckkontrollierte Beatmung	Evita 4 Evita XL Evita V500
Kombinatorische Ansätze	Wissens- und modellbasierte Beatmung	Evita V500

TABELLE 7.1 – *Mit dem Software-Framework entwickelte wissensbasierte Software-Anwendungen*

7.2 SMARTCARE

In sämtlichen Teildisziplinen der Anästhesiologie erfolgt die maschinelle Beatmung eines Patienten nach dem physikalischen Prinzip des Überdrucks. Das Beatmungsgerät appliziert aktiv mit einem positivem Druck von üblicherweise 5 bis 40 mbar einen mit Sauerstoff (21 bis 100 %) angereicherten Luftstrom für die Einatmung zum Patienten und führt die mit der passiven Ausatmung des Patienten entstandenen Gase wieder an die Umgebung ab.

Jedoch verläuft das Prinzip der Überdruckbeatmung entgegen der natürlichen, physiologischen Atmung des Menschen, die via Unterdruck gesteuert ist. Diese Umkehr der Druckverhältnisse kann wiederum zu etlichen, unerwünschten Effekten und Nebenwirkungen führen. Mediziner betrachten daher die maschinelle Beatmung grundsätzlich als schädigend und sind bestrebt, Dauer und Invasi-

vität der Beatmung so kurz und gering wie möglich zu halten. Allen mit dem Software-Framework entwickelten, für Beatmung vorgesehenen, wissensbasierten Anwendungen ist gemein, dass sie genau diese Intention als oberstes Ziel verfolgen.

Die Arbeiten an der ersten kommerzialisierten, wissensbasierten Software-Anwendung begannen bereits im Jahr 2000, basierend auf den akademischen Vorarbeiten von Prof. Brochard und seinem Team an der Universitätsklinik Henri-Mondor in Créteil, Frankreich, siehe auch Abschnitt 1.4 sowie [83]. Simultan mit dieser Portierung in ein zugelassenes Medizinprodukt startete auch die Entwicklung des Software-Frameworks selbst.

Der zugrunde liegende klinisch-therapeutische Prozess (KTP) und damit die entsprechende Zweckbestimmung war und ist die sichere Entwöhnung eines Patienten von der druckunterstützten Beatmung, engl. Pressure Support (PS). Die Entwöhnung als Schlussphase maschineller Beatmung ist äußerst zeitaufwändig und anfällig für Komplikationen. In vielen Kliniken sind medizinische Leitlinien, sogenannte „Weaning Protocols", etabliert, die die Vorgehensweise bei der Entwöhnung anleiten. Nachteilig ist, dass diese Weaning-Protokolle meist manuell vom Behandler angewendet werden müssen. Das von Brochard erarbeitete, computerisierte und klinisch evaluierte Weaning-Protokoll steuert den Einstellwert „Druckunterstützung" (PS) autonom (i. e. Vollautomatik) und wurde im Juli 2003 für das Beatmungsgerät Evita XL in Verkehr gebracht.

Zuvor gab es zu Evaluierungszwecken diverse Implementierungsvarianten für das Beatmungsgerät Evita 4, sowohl integriert als auch extern über einen Standard-PC gesteuert. Eine weitere Portierung existiert für das Nachfolgeprodukt der Evita 4, die Evita V500 sowie deren „kleinere Schwester", die Evita V300.

Abbildung 7.1 zeigt die drei therapeutischen Phasen (i. e. Subprozesse) des klinisch-therapeutischen Prozesses von SmartCare und deren wechselseitige Beziehung.

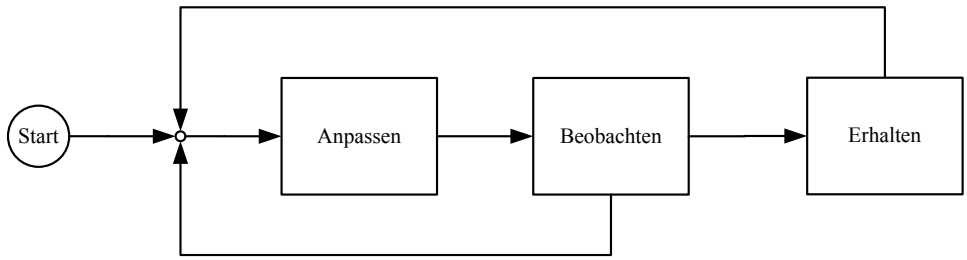

Abbildung 7.1 – *Klinisch-therapeutischer Prozess von SmartCare im Überblick*

Anpassen. Stabilisierung des Patienten innerhalb einer *respiratorischen Komfortzone* bei gleichzeitiger Reduzierung der Druckunterstützung in angemessenen Zeitabständen.

Beobachten. Durchführung einer ein- oder zweistündigen Beobachtungsphase bei minimaler Druckunterstützung als sogenannter *Spontanatemversuch*.

Erhalten. Nach der Empfehlung zur Trennung des Patienten vom Beatmungsgerät, bleibt der Patient weiterhin unter der Beobachtung des Systems und wird innerhalb der respiratorischen Komfortzone stabil gehalten.

Auffällig ist, dass der klinisch-therapeutische Prozess kein Endereignis aufweist, was bedeutet, dass er sich selbst nicht beendet. Obwohl es ein definiertes Zielkriterium gibt - der Eintritt in die therapeutische Phase *Erhalten*, bleibt es letztlich dem Klinikpersonal überlassen, explizit den Fall abzuschließen. Dies geschieht durch den Anwendungsfall *Beende Fall*, siehe Abschnitt 4.2, und soll sicherstellen, dass der Benutzer stets die finale Entscheidung treffen muss.

Mit SmartCare wurden eine Reihe klinischer Studien durchgeführt, von denen wir nachfolgend zwei vorstellen.

In einer multizentrischen, randomisierten Kontrollstudie wurden Patientensicherheit und klinische Effektivität der mit SmartCare automatisierten Entwöhnung validiert. Diese klinische Studie schloss insgesamt 144 Patienten aus Intensivstationen von fünf europäischen Universitätskliniken ein. Etwa die Hälfte der Patienten ($n = 70$) wurde gemäß der in der jeweiligen Klinik etablierten Strategie manuell entwöhnt, wohingegen die andere Hälfte der Patienten ($n = 74$) der automatisierten Therapie mit SmartCare zugeführt wurde. Im Vergleich zur konventionellen Beatmungstherapie der jeweiligen Intensivstation konnte SmartCare die Dauer der Entwöhnung von 5 auf 3 Tage, die Dauer der maschinellen Beatmung von 12 auf 7,5 Tage und die gesamte Dauer des Aufenthaltes auf der Intensivstation von 15,5 auf 12 Tage reduzieren [3].

2003 wurde der klinisch-therapeutische Prozess von SmartCare erweitert, um auch pädiatrische Patienten (Kinder und Jugendliche) von der maschinellen Beatmung entwöhnen zu können. Jouvet et al. führten dazu eine klinische Studie auf der Intensivstation einer Kinderklinik in Paris durch, die Anwendbarkeit, Toleranz und Wirksamkeit dieses SmartCare nachweisen sollte [89]. 54 Patienten zwischen ein und 17 Jahren mit einem realen Körpergewicht von über 10 kg wurden in die Studie eingeschlossen, von denen 20 mit SmartCare behandelt wurden ($n = 20$) und 34 in der Kontrollgruppe waren ($n = 34$). Das Ergebnis war eine kürzere Beatmungsdauer in der SmartCare-Gruppe ($5{,}1 \pm 4{,}2$) gegenüber ($6{,}7 \pm 11{,}5$) in der Kontrollgruppe. Es traten keine unerwünschten Zwischenfälle auf. Insgesamt schlussfolgerten die Forscher, dass SmartCare erfolgreich und wirksam für die Entwöhnung pädiatrischer Patienten eingesetzt werden konnte.

Systeme wie SmartCare sorgen auch für eine kontinuierliche Patientenbeurteilung (engl. Continuous Patient Assessment), zumindest im Rahmen ihrer jeweiligen Zweckbestimmung. SmartCare erhebt alle 10 s Daten vom Beatmungsgerät und führt alle 2 bzw. 5 min eine Inferenz durch, die zu einer Verstellung der Druckunterstützung führen kann, aber nicht muss. Zum Vergleich: ein Arzt überprüft und korrigiert die Beatmungseinstellungen üblicherweise während der Standardvisiten, also maximal zweimal täglich, wie die Studie von Hillmann et al. gezeigt hat [57].

Im Jahre 1997 brachte die Firma Hamilton Medical aus der Schweiz das Medizinprodukt *ASV* (Adaptive Support Ventilation) auf den Markt, aus dem später die Weiterentwicklung *INTELLiVENT-ASV* entstand [119]. ASV automatisiert ebenfalls die Beatmung als Closed-Loop-System und geht mit der Zweckbestimmung noch über die Entwöhnung hinaus. Auch werden weitere Beatmungsparameter autonom verstellt als die Druckunterstützung. Technisch gesehen ist ASV kein wissensbasiertes System, sondern ein klassisches Regelungssystem, dass im Wesentlichen die Otis'sche Gleichung implementiert [82]. Recherchen konnten nicht

ermitteln, ob ASV als Erzeugnis aus einem generativen Framework hervorgegangen ist.

7.3 SMART SONAR SEPSIS

Anders als beim vorangehenden Beispiel, bei dem der klinisch-therapeutische Prozess alleinig von einem individuellen Medizinexperten und seinem Team stammt, bedient sich der Smart Sonar Sepsis einer evidenzbasierten S3-Leitlinie [68]. Es geht darum, die frühzeitige Erkennung septischer Erkrankungen prozessual und computer-gestützt zu erleichtern. Zu septischen Erkrankungen zählen die Sepsis sowie deren Eskalationen der schweren Sepsis und des septischen Schocks.

Adressiert wird die Problematik der Sterblichkeit (Mortalität), die bei septischen Erkrankungen generell hoch ist und sich zudem mit fortschreitender Dauer der Nichterkennung dramatisch erhöht, wie eine aktuelle Untersuchung von Scheer et al. zeigt [61]. Der klinische Nutzen einer wissensbasierten Software-Anwendung zur möglichst frühen und sicheren Identifikation dieser Krankheitsbilder ist daher eine signifikante Reduzierung der Mortalität.

Die größte Herausforderung, die es dabei zu meistern gilt, ist, ein adäquates Zusammenspiel von Spezifität und Sensitivität zu erreichen. Sensitivität ist der Prozentsatz richtig, positiver Ergebnisse, Spezifität der Prozentsatz richtig, negativer Ergebnisse. Eine zu hohe Sensitivität oder eine zu niedrige Spezifität führen zu Fehlalarmen, wohingegen eine zu niedrige Sensitivität oder eine zu hohe Spezifität in nicht erkannten Sepsis-Situationen mündet. Adäquate Werte für beide sind vom klinischen Experten zu bestimmen, werden häufig in klinischen Studien evaluiert und sind entscheidend für Güte und Akzeptanz eines Frühwarnsystems wie dem des Smart Sonar Sepsis [59].

Das klinische Prozessbeispiel der Sepsiserkennung begleitet uns bereits seit Kapitel 2. Der aus der S3-Leitlinie abgeleitete klinisch-therapeutische Prozess ist in seiner Gesamtheit überblicksartig in Abbildung 2.4, Seite 25 dargestellt. Ein großer Vorteil, der sich aus der Verwendung einer Sx-Leitlinie ergibt, ist ein deutlich geringerer Aufwand für klinische Evaluierung und Produktzulassung.

Die wissensbasierte Software-Anwendung Smart Sonar Sepsis ist mit Java implementiert worden, arbeitet im Open-Loop-Betrieb, also als Entscheidungsunterstützung, und ist eingebettet in das Patientendaten-Managementsystem (PDMS) Integrated Care Manager (ICM) der Firma Dräger. Erste Gedanken zur Schaffung solch eines Medizinproduktes datieren auf das Jahr 2007, die Neuentwicklung selbst endete 2014 mit Erteilung des CE-Zeichens und der Inverkehrbringung.

Ein weiterer innovativer Aspekt entstand durch die Nutzung eines PDMS als Zielsystem, denn anders als bei solitären Medizingeräten wie beispielsweise einem Beatmungsgerät, laufen in einem PDMS sämtliche auf einer Intensivstation anfallenden Daten zentral zusammen. Ähnlich wie den behandelnden Personen, eröffnet die Verfügbarkeit einer vollständigen, elektronischen Patientenakte mit Anamnese, Medikationen, Laborwerten, radiologischen Aufnahmen, Vitaldaten uvm. auch wissensbasierten Systemen umfassendere Einsatzmöglichkeiten und präzisere Entscheidungen entlang des Befund-Diagnose-Ziel-Therapie-Zyklus.

Abbildung 7.2 stellt eine typische medizintechnische Topologie einer typischen Intensivstation dar, mit dem PDMS als zentrale Datenbank.

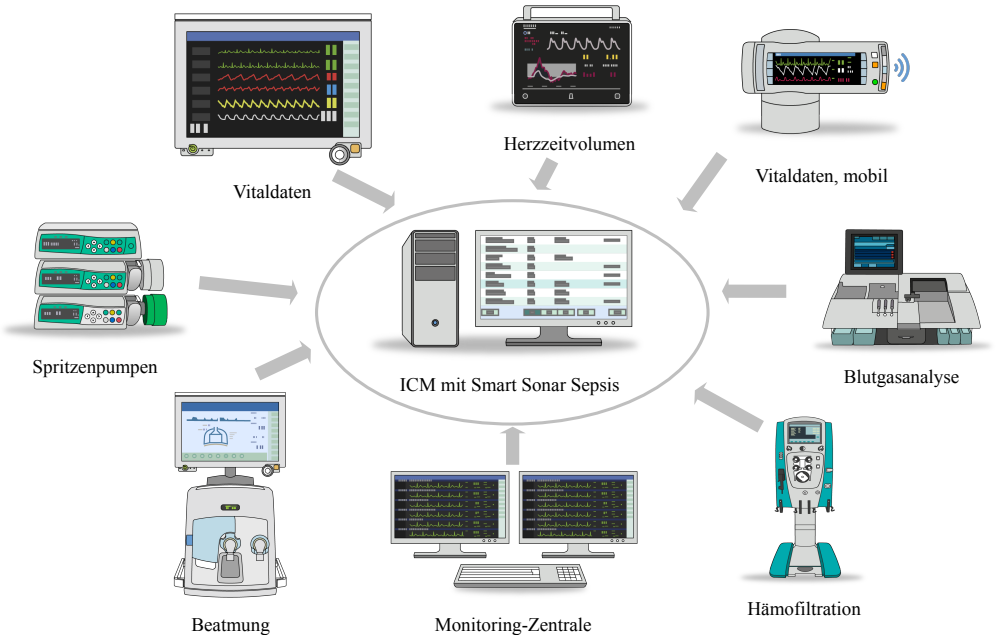

ABBILDUNG 7.2 – *Medizintechnik im Verbund mit dem PDMS ICM*

Die medizinische Fragestellung der Sepsis-Früherkennung wurde bereits 2007 von der Firma Philips aufgegriffen und mündete in einem kommerziellen Produkt mit dem Namen *ProtocolWatch*, das ebenfalls die zugrunde liegende S3-Leitlinie computerisiert und als Entscheidungsunterstützungssystem (Clinical Decision Support System) zur Verfügung steht [80]. Anders als Smart Sonar Sepsis bezieht ProtocolWatch seine Eingangsdaten für die zyklischen Inferenzen jedoch alleinig aus dem Patientenmonitor, sodass mit den dort verfügbaren Vitaldaten lediglich ein Teilbereich der S3-Leitlinie abgedeckt werden kann und auch die Erweiterungsmöglichkeiten des Systems limitiert sind.

7.4 SMART VENTILATION CONTROL

Aus den Kernprinzipien und Grundkonzepten des SmartCare sowie einem erwarteten klinischen Mehrwert, entstanden 2007 die Ideen zum *Smart Ventilation Control (SVC)*.

Der Abschluss einer medizinischen Operation, die sogenannte *Ausleitung*, ist äußerst anspruchsvoll hinsichtlich Personalaufwand, Interventionen an Patient und Medizintechnik und stellt eine enorme Belastung für das Klinikpersonal dar. Es geht darum, in möglichst kurzer Zeit die Auswaschung der Narkosegase aus dem Organismus des Patienten zu bewerkstelligen und gleichzeitig den operativen Eingriff so punktgenau zu beenden, dass der Patient zu Bewusstsein kommt, ohne jedoch unnötig Schmerzen zu erleiden.

Wenn eine wissensbasierte Software-Anwendung während der Ausleitung die Beatmung des Patienten übernehmen kann, so entlastet sie damit alle Beteiligten - inklusive des Patienten - ganz erheblich. Das war die anfängliche Intention in der Diskussion mit dem medizinischen Experten Prof. Hörmann, der zu der Zeit (2009) als Direktor der Intensivmedizin und Anästhesie des Universitätsklinikum Innsbruck, Österreich, mit seinem Team zu Themen der automatisierten Therapieführung forschte.

Im Zuge der Projektinitiierung und Ermittlung der Produktanforderungen wurde dann die Zweckbestimmung der wissensbasierten Software-Anwendung auf die gesamte, maschinelle Beatmung während einer Narkose ausgeweitet. Damit erhöht sich der klinische Mehrwert dieses Medizinproduktes neben der Entlastung und Güte der Ausleitung um eine lungenprotektive Beatmung von Anbeginn der Narkose bis zum Abschluss der Operation [51]. Besonderer Fokus liegt auf der Spontanatmung (SA) des Patienten, die je nach klinischem und physiologischen Kontext unterdrückt, zugelassen oder angeregt werden kann.

Ebenso wie bei SmartCare ist die übergeordnete Therapiestrategie auch bei SVC, den Patienten durch geeignete Einstellung relevanter Parameter in eine respiratorische Komfortzone zu bringen und dort zu halten. Hierfür titriert das System zyklisch die Beatmungseinstellungen Atemfrequenz, Druckunterstützung, Einatemdruck, Einatemzeit und Einatemempfindlichkeit, um die beatmungsspezifischen Messwerte endtidales Kohlendioxid und Tidalvolumen in vorgegebene Zielbereiche zu führen. Innerhalb der respiratorischen Komfortzone ist der Patient physiologisch optimal beatmet.

Abbildung 7.3 stellt den klinisch-therapeutischen Prozess im Gesamtüberblick als Flussdiagramm dar.

Nach Eingabe patientenspezifischer Daten können bis zu sechs Subprozesse die Beatmung autonom steuern. Deren Abfolge ergibt sich primär aus der Angabe eines Therapieziels durch den Benutzer und sekundär durch den klinischen Kontext während der Prozessausführung. Die in den Subprozessen umgesetzten Therapieziele sind:

Anpassung. Die Auswertung mehrerer Einstell- und Messwerte stellt fest, ob der jeweilige Patient für eine Anwendung von SVC geeignet ist.

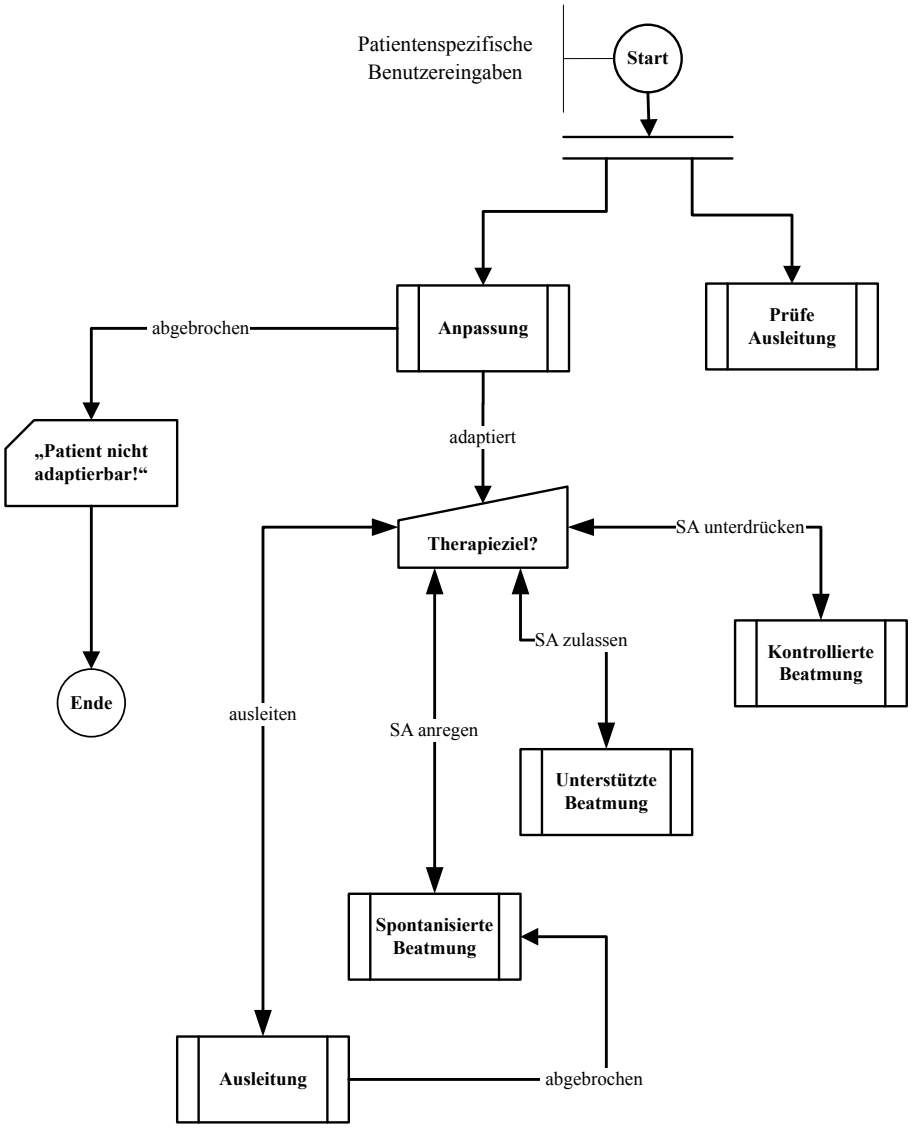

ABBILDUNG 7.3 – *Klinisch-therapeutischer Prozess des Smart Ventilation Control*

Kontrollierte Beatmung. Der Patient soll vollumfänglich maschinell beatmet werden und keinerlei Eigenatmung haben. Dies ist Voraussetzung für chirurgische Eingriffe im Bauchraum, die nicht durch Zwerchfellaktivitäten des Patienten gestört werden dürfen.

Unterstützte Beatmung. Die Spontanatmung des Patienten darf koexistieren mit maschinellen Aktivitäten des Beatmungsgerätes.

Spontanisierte Beatmung. Die Bemühungen des Patienten, wieder selbstständig zu atmen sollen nicht nur unterstützt, sondern auch trainiert werden.

Ausleitung. Wenn sämtliche Narkosegase durch Abatmung den Patienten verlassen haben, kann die sukzessive Beendigung der Narkosebeatmung sowie der Abschluss der Operation beginnen.

Parallel dazu ermittelt der Subprozess *Prüfe Ausleitung* die Erfüllung von Voraussetzungen für die Schlussphase der Narkose. Weiterhin neu mit Smart Ventilation Control ist der autonome Wechsel zwischen Beatmungsformen während der Ausführung des klinisch-therapeutischen Prozesses.

Derzeitiges Zielsystem für Smart Ventilation Control ist der Anästhesie-Arbeitsplatz Zeus IE der Firma Dräger. Die Firmware des Zeus IE ist in der Sprache „C" für das Echtzeitbetriebssystem *vxWorks* programmiert. Als Laufzeitsystem zur Ausführung des Java-Bytecodes der wissensbasierten Software-Anwendung ist die kommerzielle Java Virtual Machine (JVM) *Personal JWorks* der Firma WindRiver im Einsatz. Die Anbindung des Zielsystems erfolgte mit dem Java Native Interface (JNI).

Eine erste klinische Evaluierung mit Smart Ventilation Control fand 2015 in den Universitätskliniken Kiel und St. Pölten (Österreich) statt. Das exakte Vorgehen für diese klinische Studie ist in Schädler et al. beschrieben [63]. Primäre Fragestellung war, ob Smart Ventilation Control eine sichere und wirksame Narkosebeatmung gewährleisten kann. Dazu wurden zuvor definierte, sogenannte unerwünschte Ereignisse (engl. Adverse Events) wie beispielsweise Hypo- oder Hyperventilation oder anhaltender Atemstillstand (Apnoe) herangezogen. In diese Studie wurden 100 Patienten ($n = 100$) mit einem Durchschnittsalter von 64,99 Jahren eingeschlossen. Während der Studie traten 18 unerwünschte Ereignisse auf ($n = 18$), die jedoch von Smart Ventilation Control adäquat bereinigt werden konnten, woraus die Forscher schlussfolgerten, dass mit SVC eine sichere und lungenprotektive, automatisierte Beatmung im Rahmen einer Allgemeinanästhesie gewährleistet ist [73].

Für die Realisierung einer automatisierten Narkosebeatmung im Allgemeinen und für Teile des klinisch-therapeutischen Prozesses im Besonderen haben wir insgesamt drei Patente angemeldet, die 2017 auch erteilt wurden [26], [24], [25].

7.5 MASCHINELLE MASKENBEATMUNG

Das Entwicklungsprojekt *CG PS-NIV* beschäftigte sich ebenfalls mit der maschinellen Beatmung, vornehmlich auf der Intensivstation. PS steht für Pressure Support und bezeichnet die Druckunterstützung der Spontanatmung eines Patienten. NIV ist die Abkürzung für Non-Invasive Ventilation (Nicht-invasive Beatmung) und meint damit eine Maskenbeatmung, bei der der Patient nicht via Luftröhre, sondern mittels einer Maske mit dem Beatmungsgerät verbunden ist. Die Maskenbeatmung schließt sich häufig einer konventionellen Beatmung an, wenn beispielsweise ein Patient noch nicht vollständig entwöhnt werden kann oder soll oder aber aus anderen klinischen Gründen eine Langzeitbeatmung bei nur schwachem Spontanantrieb indiziert ist.

Abbildung 7.4 zeigt die drei gebräuchlichsten Formen von Beatmungsmasken, die entweder nur die Nase bedecken (nasal), Mund und Nase (oronasal) oder das gesamte Gesicht (total-face).

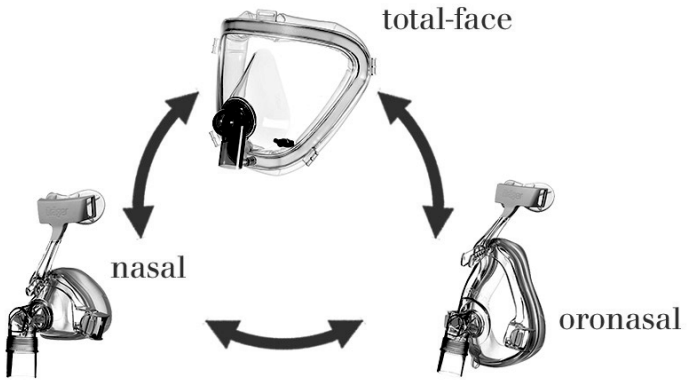

ABBILDUNG 7.4 – *Gängige Formen von Beatmungsmasken*

NIV bietet eine Reihe von Vorteilen, die Oczenski [51] wie folgt zusammenfasst:

- bessere Toleranz durch den Patienten
- Kommunikationsfähigkeit bleibt erhalten
- Orale Ernährung ist möglich
- natürliche Anfeuchtung der Atemluft
- bessere Mobilisation der Patienten
- kürzere Beatmungsdauer, kürzerer Aufenthalt auf der Intensivstation

Trotz der überwiegenden Vorteile gestaltet sich die Maskenbeatmung in der klinischen Praxis recht kompliziert, sowohl für das Pflegepersonal, als auch technisch für das jeweilige Beatmungsgerät. Individualität und Variabilität der Gesichtsformen der Patienten bedingen quasi inhärent, das die jeweilige Maske selten präzise und passgenau sitzt. Dies führt zu Leckagen im gesamten Beatmungssystem, die wiederum in der Folge in Fehlalarmen oder gar Asynchronitäten zwischen Patient und Beatmungsgerät münden können. Die technische Herausforderung für ein Beatmungsgerät, das NIV unterstützt, besteht also in einem adäquaten Management von Leckagen.

Aufbauend auf den Kernideen des SmartCare, insbesondere der respiratorischen Komfortzone, lieferte Prof. Jolliet im Jahre 2004 mit seinem Team der Universitätsklinik Genf (Schweiz) einen klinisch-therapeutischen Prozess zur teilautomatisierten Maskenbeatmung. Die daraus resultierende wissensbasierte Software-Anwendung liegt in zwei Implementierungen vor: CE-zugelassen für das Beatmungsgerät Evita XL zur Durchführung einer klinischen Studie, die Sicherheit und klinischen Nutzen zeigen sollte. Eine weitere Implementierung erfolgte für das Beatmungsgerät Evita V500, ebenfalls mit CE-Zulassung, jedoch ohne Vermarktung. Ähnlich dem SmartCare, steuert CG PS-NIV ausschließlich den Beatmungsparameter Druckunterstützung (PS) im Modus Vollautomatik.

Oben erwähnte klinische Studie fand 2004 im Universitätsklinikum Genf (Schweiz) statt und wurde 2006 publiziert [53]. Ziel war es, die generelle Eignung zu prüfen, indem bewertet wird, wie gut die Patienten von CG PS-NIV in der

respiratorischen Komfortzone gehalten werden können. Zwanzig Patienten (n = 20) mit akutem Lungenversagen wurden in die Studie eingeschlossen. Die Auswertung der relevanten physiologischen Parameter ergab, das CG PS-NIV geeignet ist für die teilautomatisierende Maskenbeatmung.

Für die Realisierung der teilautomatisierten Maskenbeatmung inklusive des Managements von Leckagen und die Überwachung der Beatmungsparameter Tidalvolumen und endtidales Kohlendioxid haben wir ein Patent angemeldet, das 2013 auch erteilt wurde [23].

7.6 DRUCKKONTROLLIERTE BEATMUNG

Während SmartCare mit der Entwöhnung eine Teilautomatisierung der intensivmedizinischen Beatmung adressiert, gehen die computerisierten, klinisch-therapeutischen Prozesse CG APRV und CG BIPAP einen Schritt weiter und haben die automatisierte Therapieführung für die gesamte maschinelle Beatmung zum Ziel. Dazu ist nicht nur eine autonome Steuerung aller verfügbaren Beatmungseinsteller erforderlich, sondern auch die alternierende Anwendung verschiedener Beatmungsformen, genauer: druckunterstützte und druckkontrollierte Beatmung.

APRV (Airway Pressure Release Ventilation) und auch BIPAP (Biphasic Positive Airway Pressure) zählen beide zu den druckkontrollierten Beatmungsformen, was bedeutet, das die maschinelle Beatmungssteuerung primär über den inspiratorischen Beatmungsdruck erfolgt. Bei erkannten Eigenatembemühungen des Patienten muss das Beatmungsgerät zudem eine Mischform ermöglichen, bei der sowohl eine rein maschinelle als auch die Spontanatmung unterstützende Beatmung erfolgt. Konsequenterweise nutzen beide wissensbasierte Software-Anwendungen nennenswerte Anteile des dem SmartCare zugrundeliegenden klinisch-therapeutischen Prozess. Sie sind in mehreren Ausprägungen sowohl zur Entscheidungsunterstützung als auch in der Vollautomatik implementiert worden und steuern autonom bis zu sieben Parameter, inklusive dem Wechsel zwischen Beatmungsformen.

Neuartig für beide Systeme ist, dass sie erstmalig auch die Sauerstoffkonzentration im Einatemgas (F_iO_2) autonom verstellen können.

Eine präklinische Studie mit CG BIPAP in Verbindung mit einem Human Patient Simulator fand 2010 im Universitätsklinikum Kiel statt, siehe Abschnitt 6.6.

2014 publizierten Schädler et al. die Ergebnisse einer klinischen Studie von 19 Patienten (n = 19), mit der untersucht wurde, ob CG BIPAP geeignet ist für die automatische Entwöhnung und dabei die Invasivität der Beatmung (engl. respiratory load) reduzieren kann [1]. Die Auswertung der aufgezeichneten physiologischen Parameter bestätigten diese Hypothese.

CG BIPAP ist vergleichbar mit dem weiter oben angeführten INTELLiVENT-ASV der Firma Hamiton Medical, mit dem Unterschied, dass CG BIPAP ein wissenbasiertes und kein regelungstechnisches System ist und zudem einen größeren Teil der Beatmungstherapie abdeckt.

7.7 KOMBINATORISCHE ANSÄTZE

Im BMBF-Forschungsprogramm „IKT 2020 - Forschung für Innovationen" wurde das Grundkonzept der prozessorientierten, wissensbasierten Systeme in der Anästhesiologie in einen signifikant größeren Kontext gestellt und umgesetzt.

Kernidee war, mit Hilfe mathematischer Modellierung physiologischer Systeme die Qualität essenzieller Parameter zu verbessern sowie weitere Parameter zu berechnen. Dieser Ansatz führte zur Kombination wissensbasierter mit modellbasierten Technologien, was dem Vorhaben den Projektnamen „Wissens- und modellbasierte Therapiesteuerung in der Medizin - von der komplizierten Beatmungseinstellung zur individuellen strategischen Therapieführung (WiM-Vent)" gab.

Das Forschungsprojekt startete im Januar 2011 und endete im Dezember 2013. Das Konsortium bestand aus folgenden Partnerschaften, die jeweils ihre wissenschaftlichen Vorarbeiten einbrachten.

- Albert-Ludwigs-Universität Freiburg (Prof. Guttmann)
- Hochschule Furtwangen (Prof. Möller)
- Christian-Albrechts-Universität zu Kiel (Prof. Weiler)
- Julius-Maximilians-Universität Würzburg (Prof. Puppe)
- Drägerwerk AG & Co. KGaA

Den wissensverarbeitenden Teil lieferte der CG BIPAP (Abschnitt 7.6), die physiologischen Modelle basierten u. a. auf den Arbeiten von Kretschmer et al. [78]. Technologisch erfolgte die „Hochzeit" der beiden Methodiken über die Implementierung eines weiteren Slots *PMOD* in die basale Systemarchitektur, siehe Abbildung 4.2, Seite 46. Die bidirektionale Datenanbindung an ein Patientendaten-Managementsystem (PDMS) mit dem Ziel, weitere Einsatzspektren und Anwendungsszenarien zu eröffnen, wurde ebenfalls durch einen zusätzlichen Slot mit Namen *PDMS* implementiert.

7.8 WEITERE BEISPIELANWENDUNGEN

Einige mit dem Software-Framework entwickelte Anwendungen waren von vornherein für die Untersuchung wissenschaftlicher Fragestellungen und spezifischer Forschungsgebiete konzipiert. Im Folgenden werden sie aufgeführt.

Proportionale, druckunterstützte Beatmung. Dem übergeordneten Anspruch, für möglichst viele Formen der maschinellen Beatmung auf der Intensivstation durch geeignete wissensbasierte Software-Anwendungen eine (Teil)Automatisierung zu bieten, kommt auch der *CG PPS* nach. Die hier betrachtete Beatmungsform unterstützt die Spontanatmung eines Patienten proportional zu dessen Einatembemühung, daher die Bezeichnung PPS für Proportional Pressure Support. Auch hier lässt sich die therapeutische Strategie des SmartCare wieder- und weiterverwenden.

In Mersmann et al. [9] ist diese detailliert beschrieben. Sie kann somit als Lastenheft für die Entwicklung einer automatisierenden, wissensbasierten Software-Anwendung für Beatmungsgeräte Verwendung finden.

Abbildung 7.5 zeigt den klinisch-therapeutischen Prozess, der als Design-Input für dieses Entwicklungsprojekt diente. Im Gegensatz zu SmartCare (Abschnitt 7.2) gibt es hier den zusätzlichen Teilprozess *Übernahme*, der vor Therapiebeginn die Eignung eines Patienten überprüft und im negativen Fall zur Terminierung des ganzen Prozesses führt. Alle anderen Subprozesse ähneln denen des SmartCare.

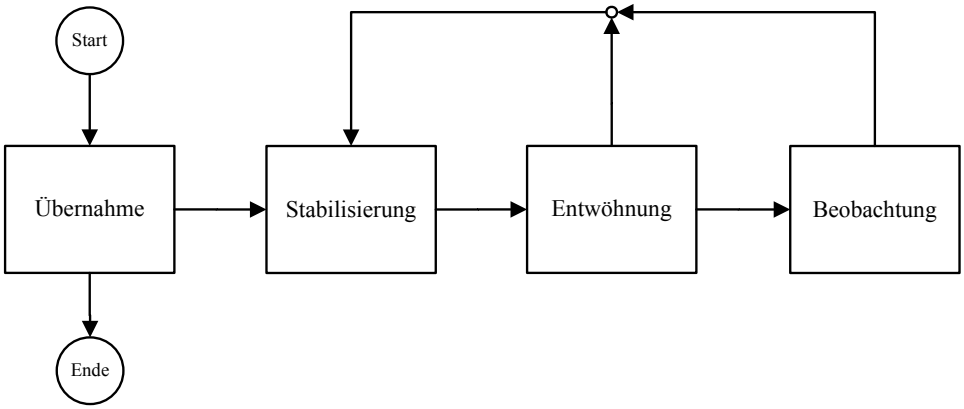

ABBILDUNG 7.5 – *Klinisch-therapeutischer Prozess des CG-PPS im Überblick*

Glukosemanagement. Die Ernährung und speziell die künstliche Ernährung von Patienten stellt eine wichtige Komponente in der Intensivmedizin und gleichzeitig eine große Herausforderung für das Personal dar. Zur Vereinfachung und Qualitätssicherung dieser Thematik haben diverse Organisationen und Fachverbände in Deutschland die S2k-Leitlinie „Klinische Ernährung in der Intensivmedizin" entwickelt und etabliert [71]. Ein Schwerpunkt dieser medizinischen Leitlinie ist die Überwachung des Glukosespiegels (Blutzucker), mit dem wesentlichen Ziel, eine Hypoglykämie (Unterzuckung) und eine Hyperglykämie (Überzuckerung) durch frühzeitige Erkennung zu vermeiden. Aus der Leitlinie lässt sich ein prozessuales Vorgehen ableiten und als klinisch-therapeutischer Prozess computerisieren.

Die studentische Abschlussarbeit von Plachetka [34] beschäftigte sich mit dieser Aufgabe und führte den resultierenden klinisch-therapeutischen Prozess über die CKEP-Entwicklungsphasen Identifikation, Extraktion, Interpretation und Operationalisierung (siehe Abschnitt 4.6) bis hin zur initialen Modellierung in KnowWE. Innerhalb von KnowWE konnte dieser klinisch-therapeutische Prozess auch ausgeführt, vorgeführt und explorativ getestet werden. Eine darauf bauende wissensbasierte Software-Anwendung könnte in einem PDMS dem Klinikpersonal als Entscheidungsunterstützung dienen.

Delirmanagement. Einen sehr ähnlichen Ansatz wie die CG Glukose verfolgt die CG Delir, mit der das immer prekärer werdende Phänomen des Delir auf der Intensivstation entschärft werden könnte. Das häufig auch als „Durchgangssyndrom" bezeichnete Delir kennzeichnet einen kritischen, akuten Verwirrtheitszustand des Patienten. Meist liegt eine gleichzeitige Störung kognitiver Dispositionen

vor, die Bewusstsein und Aufmerksamkeit, Wahrnehmung, Denken, Gedächtnis, Psychomotorik, Emotionalität und Schlaf-Wach-Rhythmus betreffen kann [93]. Ein Delir verzögert nicht nur die Genesung, sondern kann in schweren Fällen mitunter zu aggressivem und gewalttätigen Verhalten von Patienten führen.

Die S3-Leitlinie *Analgesie, Sedierung und Delirmanagement in der Intensivmedizin (DAS-Leitlinie)* [69] bietet auch hier die Möglichkeit, einen entsprechenden klinisch-therapeutischen Prozess als Entscheidungsunterstützungssystem hervor zu bringen. Potenzielles Zielsystem wäre auch hier ein Patientendaten-Managementsystem.

Wir haben in diesem Kapitel unterschiedliche, wissensbasierte Software-Anwendungen vorgestellt, die mit dem Software-Framework entwickelt und in Betrieb genommen wurden und sind dabei auch auf klinische Besonderheiten und Herausforderungen eingegangen. Sieben davon waren reguläre Produktentwicklungen und drei dienten wissenschaftlichen Forschungszwecken.

Die erzielte Vielfältigkeit der computerisierten klinisch-therapeutischen Prozesse auf der einen Seite und die der Zielsysteme auf der anderen Seite belegen die Wirksamkeit und den Mehrwert unserer zwei Prinzipien und drei Paradigmen. Explizit erkennbar ist dies beim Paradigma der zwei Freiheitsgrade, wohingegen die Paradigmen zweiphasiger Nutzungszyklus und hybrides Vorgehensmodell ihre Vorteile während der Entwicklung der wissensbasierten Software-Anwendungen zeigen.

Die ökonomischen Vorteile lassen sich am Beispiel SmartCare quantifizieren. So hat eine Gemeinschaftsstudie der Spectaris 2006, aufbauend auf den Ergebnissen der klinischen Studie von Lellouche et al. [3], ermittelt, dass mit der Nutzung von SmartCare in allen Intensivstationen Deutschlands (i. e. 11.000 Patientenbetten) das Einsparpotenzial aus einer Reduktion der Sach- und Personalkosten in Deutschland rund 648 Millionen Euro pro Jahr beträgt [42]. Zusammen mit den Ergebnissen der Lellouche-Studie werden alle vier Ziele des Quadruple-Aim (Abschnitt 1.1) erreicht: Gesundheit der Bevölkerung, Pro-Kopf-Kosten, Behandlungsqualität und Zufriedenheit des Klinikpersonals. Und auch die allgemeinen Vorteile medizinischer Leitlinien aus Abschnitt 1.2 (Seite 4) können nachweislich genutzt werden.

KAPITEL 8

PROZESSMANAGEMENT FÜR MEDIZINISCHE LEITLINIEN

Primäres Konzept dieser Arbeit ist es, komplexe Prozesse im klinisch-therapeutischen Umfeld der Anästhesiologie systematisch zu computerisieren und dem Klinikpersonal als wissensbasierte Software-Anwendungen zur Verfügung zu stellen. Die hier erarbeitete Lösung eines Software-Framework ist die erste Generation ihrer Art.

Im Bereich der Unternehmensführung und dort insbesondere in der Organisationsentwicklung hat sich seit den 20-er Jahren des letzten Jahrhunderts der Industriestandard des Geschäftsprozessmanagements, oder kurz Prozessmanagement, bewährt.

In diesem Kapitel liegt das Hauptaugenmerk auf der Beantwortung der Frage, welche Aspekte des modernen Prozessmanagement sich für wissensbasierte Systeme in der Anästhesiologie vorteilhaft nutzen lassen.

Es geht also im Wesentlichen um die Substitution proprietär entwickelter Bausteine der hier vorgestellten Lösung durch Bausteine des konventionellen, standardisierten Prozessmanagement und somit zu einer zweiten Generation des Software-Frameworks zu kommen.

Nach einer kompakten Einführung in die grundlegenden Prinzipien und Aufgaben des Prozessmanagements richten wir den Fokus auf zwei zentrale Themenbereiche: die Modellierung von Prozessen sowie deren Ausführung. Beide Aspekte stehen in engem Zusammenhang mit der zuvor dargestellten Gesamtlösung und bieten daher ein besonders großes Potenzial für eine gezielte Modernisierung.

Sowohl das allgemeine als auch das klinische Prozessbeispiel aus Kapitel 2 werden wiederverwendet, mit Methoden des modernen Prozessmanagement modelliert und grafisch dargestellt. Mit der Übertragung dieser Modellierung auf die gegebene Problemstellung, also: dem klinisch-therapeutischen Einsatz in der Anästhesiologie, sowie einem Fazit schließt dieses Kapitel.

8.1 HINTERGRUND

Die ersten Ideen und Konzepte zum Geschäftsprozessmanagement, engl. Business Process Management (BPM), entstanden bereits in den 1920er Jahren, als Unternehmen begannen, sich intensiver mit der Organisation und Optimierung ihrer betrieblichen Abläufe zu beschäftigen. In den folgenden Jahrzehnten entwickelte sich das Prozessmanagement weiter, wobei verschiedene Ansätze und Methoden entstanden.

Die European Association of Business Process Management (EABPM) definiert Prozessmanagement wie folgt [28]:

Definition PROZESSMANAGEMENT

> Business Process Management (BPM) ist eine Managementdisziplin, mit der Strategien und Ziele einer Organisation und die Erwartungen und Bedürfnisse von Kunden miteinander verbunden werden, indem Ende-zu-Ende-Prozesse in den Mittelpunkt gestellt werden. BPM umfasst Strategien, Ziele, Kultur, Organisationsstrukturen, Rollen, Grundsätze, Regeln, Methoden und IT-Werkzeuge, um a) Ende-zu-Ende-Prozesse zu analysieren, zu entwerfen, einzuführen, zu steuern und kontinuierlich zu verbessern und b) eine Prozess-Governance aufzubauen.

Im Mittelpunkt des Prozessmanagement steht also immer der Prozess, so wie wir ihn in Abschnitt 2.2 definiert haben, hier jedoch zusätzlich erweitert um den sogenannten Ende-zu-Ende-Ansatz (E2E). Dieser fordert eine stringente Abfolge von Prozessschritten, die vom Anfang bis zum Ende einer bestimmten Geschäftsaktivität oder eines Arbeitsablaufes reichen. Es soll sichergestellt werden, dass alle Schritte, Aktivitäten und Beteiligten innerhalb eines Prozesses vollständig integriert und beschrieben sind, um das gewünschte Ergebnis effizient und effektiv zu erreichen. Ein Ende-zu-Ende-Prozessansatz zielt darauf ab, die gesamte Wertschöpfungskette zu optimieren, anstatt einzelne Komponenten isoliert zu betrachten. Krankenhäuser, Kliniken und ärztliche Praxen sind ebenfalls wirtschaftliche Unternehmungen, die sich mehr und mehr über Geschäftsprozesse steuern, um rentabel ihre gesetzten Unternehmensziele erreichen zu können.

Für klinisch-administrative Prozesse (siehe Abschnitt 2.2.2) hat sich das Geschäftsprozessmanagement mitsamt dem darauf spezialisierten Business Process Management System (BPMS) bereits vielfach etabliert. Doch auch klinisch-therapeutische Prozesse lassen sich per Definition als E2E Geschäftsprozesse begreifen. Das E2E-Kriterium trägt sogar zu einer zusätzlichen Qualitätssteigerung bei, weil damit die Geschlossenheit und Vollständigkeit einer Prozessspezifikation systemisch prüfbar wird.

Warum also sollten Methoden aus dem modernen Geschäftsprozessmanagement nicht ebenso für klinisch-therapeutische Prozesse wirksam oder gar effizient sein? Mit geeigneten Antworten auf diese Frage kann die zweite Generation des Software-Frameworks entstehen.

8.2 MODELLIERUNG VON PROZESSEN

Die Modellierung von Geschäftsprozessen - im Kontext des Prozessmanagement auch *Digitalisierung von Prozessen* genannt - ist die erste Stufe eines modernen Prozessmanagements und Voraussetzung für die Prozessausführung, die zweite Stufe. Der weltweite Industriestandard der Business Process Model and Notation (BPMN) regelt die Modellierung von Prozessen und ist mit der ISO 19510 in der aktuellen Version 2.0.1 spezifiziert [109]. Sie bietet eine standardisierte, visuelle Darstellung, mit der sowohl einfache als auch komplexe Geschäftsprozesse beschrieben werden können. BPMN zielt darauf ab, eine gemeinsame Sprache zwischen den verschiedenen Interessengruppen in einem Unternehmen zu schaffen und ist ein wichtiges Instrument im Bereich des Geschäftsprozessmanagements, das dazu beiträgt, Prozesse klar und präzise zu dokumentieren, zu analysieren und zu optimieren. Sie wurde erstmals im Jahr 2004 veröffentlicht.

In der derzeitigen Ausprägung bietet die BPMN eine Fülle von Modellierungselementen für die folgenden Informationskategorien:

Aktivitäten sind Prozessschritte wie Aufgaben, Transaktionen, Teilprozesse oder Aufrufaktivitäten, die via Sequenz-, Standard- oder bedingtem Fluss verbunden werden können.

Konversationen definieren einen mehrfachen, logisch zusammengehörigen Nachrichtenaustausch, der Teilnehmer über einen Konversationslink verbindet.

Choreographien sind Diagramme, die den Austausch von Nachrichten zwischen verschiedenen Teilnehmern eines Geschäftsprozesses darstellen, wobei der Fokus auf der Interaktion und Kommunikation liegt.

Gateways beschreiben Verzweigungen im Kontrollfluss, die Entscheidungen via logischer Verzweigungsbedingungen ermöglichen. Die wichtigsten Gateways sind das exklusive Gateway (XOR), das inklusive Gateway (OR) und das parallele Gateway (AND).

Daten repräsentieren jegliche Art von Information, die im Prozess verwendet und/oder erzeugt werden. Sie können einem Datenspeicher entnommen oder in diesem abgelegt werden.

Ereignisse stellen bestimmte Vorfälle oder Zustände innerhalb eines Prozesses dar. Es gibt Start-, Zwischen- und Endereignisse. Wichtige Ereignistypen sind Blanko (untypisiert), Nachrichten, Timer, Fehler, Abbruch und Signale.

Swimlanes repräsentieren Verantwortlichkeiten für Aktivitäten. Eine Swimlane kann eine Organisation, eine Rolle oder ein System sein und strukturiert Verantwortlichkeiten.

All diese Modellierungselemente können genutzt werden, um in sogenannten *Kollaborationsdiagrammen* die entsprechenden Prozesse gemäß der erforderlichen Angaben aus Abschnitt 2.2 vollständig, grafisch zu beschreiben.

Eine gute Übersicht aller Modellierungselemente gibt das BPMN-Poster der Berliner BPM Offensive [70]. Ergänzend zur BPMN ist die Decision Model and Notation (DMN), ein Standard, um Entscheidungsregeln und Entscheidungslogik

in einem strukturierten und leicht verständlichen Format zu modellieren [45]. Mit der DMN lassen sich regelbasierte Problemlösungsstrategien (Abschnitt 3.4) umsetzen.

IT-Systeme für das Geschäftsprozessmanagement existieren seit den frühen 1990er Jahren und werden unter dem Begriff *Business Process Management System (BPMS)* geführt. Eine umfassende Übersicht findet sich beispielsweise in der Marktstudie von Drawehn et al. [33]. Die zentralen Funktionalitäten eines BPMS sind

- Prozessmodellierung (BPMN, DMN)
- Prozessausführung und -automatisierung
- Überwachung und Analyse
- Integration mit anderen IT-Systemen
- Optimierung
- Kollaboration

Die neueste Generation von BPMS werden als „Software as a Service (SaaS)" cloud-basiert vertrieben, nutzen ausschließlich Webtechnologien und bieten damit ein breites Spektrum für Erweiterbarkeit und Integration. Prominente kommerzielle Vertreter dieser Generation sind BIC der Firma GBTEC, Signavio der Firma SAP sowie iGrafix der gleichnamigen Firma. Frei verfügbar als Open-Source-Version ist das unter der Apache Lizenz 2.0 veröffentlichte BPMS von Camunda.

8.2.1 BEISPIELPROZESS, ALLGEMEIN

Abbildung 8.1 greift das in Abschnitt 2.2.3 vorgestellte Beispiel des Prozesses zur Entwicklung medizinischer Software auf und gibt dessen semantisch äquivalente BPMN-Modellierung in einem Kollaborationsdiagramm wieder.

Hauptakteur ist das *Development Team*, das gemäß BPMN als Swimlane dargestellt ist. Innerhalb dieser Swimlane ist der gesamte Prozess modelliert, inhaltlich äquivalent zu der Modellierung als Programmablaufplan in Abbildung 2.3, Seite 24. Verwendet werden die BPMN-Modellierungselemente Start- und Endereignis (beide untypisiert), Subprozesse (*Classify Software*), Aktivitäten (z. B. *Specify Software Requirements*) sowie inklusive Gateways (Raute mit Kreis) zur Modellierung einer Schleife für Änderungszyklen. Zusätzlich zur PAP-Modellierung und der Prozessspezifikation aus Abschnitt 2.2 folgend, wurden Eingänge und -ausgänge für Dokumente (z. B. *Product Requirements*) und Datenspeicher (*Software System*) angegeben.

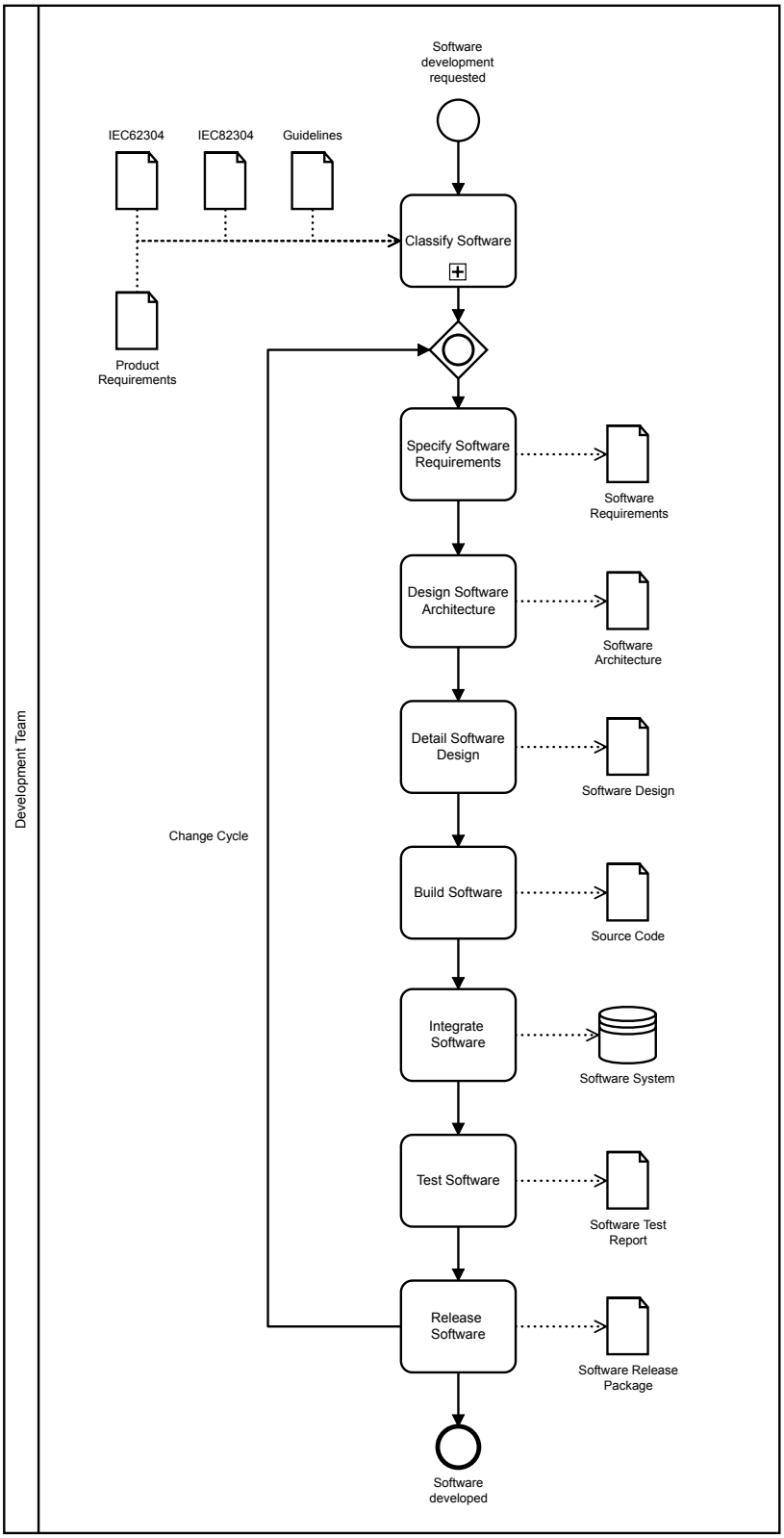

Abbildung 8.1 – *Entwicklung medizinischer Software in BPMN*

8.2.2 BEISPIELPROZESS, KLINISCH-THERAPEUTISCH

Das in Abschnitt 2.2.4 eingeführte Beispiel für einen klinisch-therapeutischen Prozess zur Erkennung und zum Management septischer Erkrankungen, als Programmablaufplan illustriert in Abbildung 2.4, Seite 25, ist Grundlage und Teil des Lastenheftes für die Entwicklung der wissensbasierten Software-Anwendung *Smart Sonar Sepsis*, siehe Abschnitt 7.3.

Die entsprechende, äquivalente Modellierung in BPMN als Kollaborationsdiagramm zeigt Abbildung 8.2. Hier ist der Hauptakteur ein *Healthcare Provider* - zu deutsch etwa: Gesundheitsdienstleister, der wieder als Swimlane modelliert wurde. Zusätzlich zum Einsatz kommen die Modellierungselemente Zwischenereignis Timer (*Wait 60sec*), Zwischenereignis Nachricht (*Sepsis detected*) sowie weiter zu detaillierende Subprozesse (Aktivitäten mit Pluszeichen). Letztere bilden - wie schon im korrespondierenden PAP - die drei Eskalationen zur Erkennung (*Monitor for ...*) und zur Behandlung (*Handle ...*) einer Sepsis, einer schweren Sepsis und eines septischen Schocks ab. Die parallele Ausführung der Subprozesse ist durch ein Parallel-Gateway (Raute mit Pluszeichen) modelliert, die vor der nächsten Inferenz - also nach Ablauf von 60 s - mit einem inklusiven Gateway (Raute mit Kreis) wieder zusammengeführt werden.

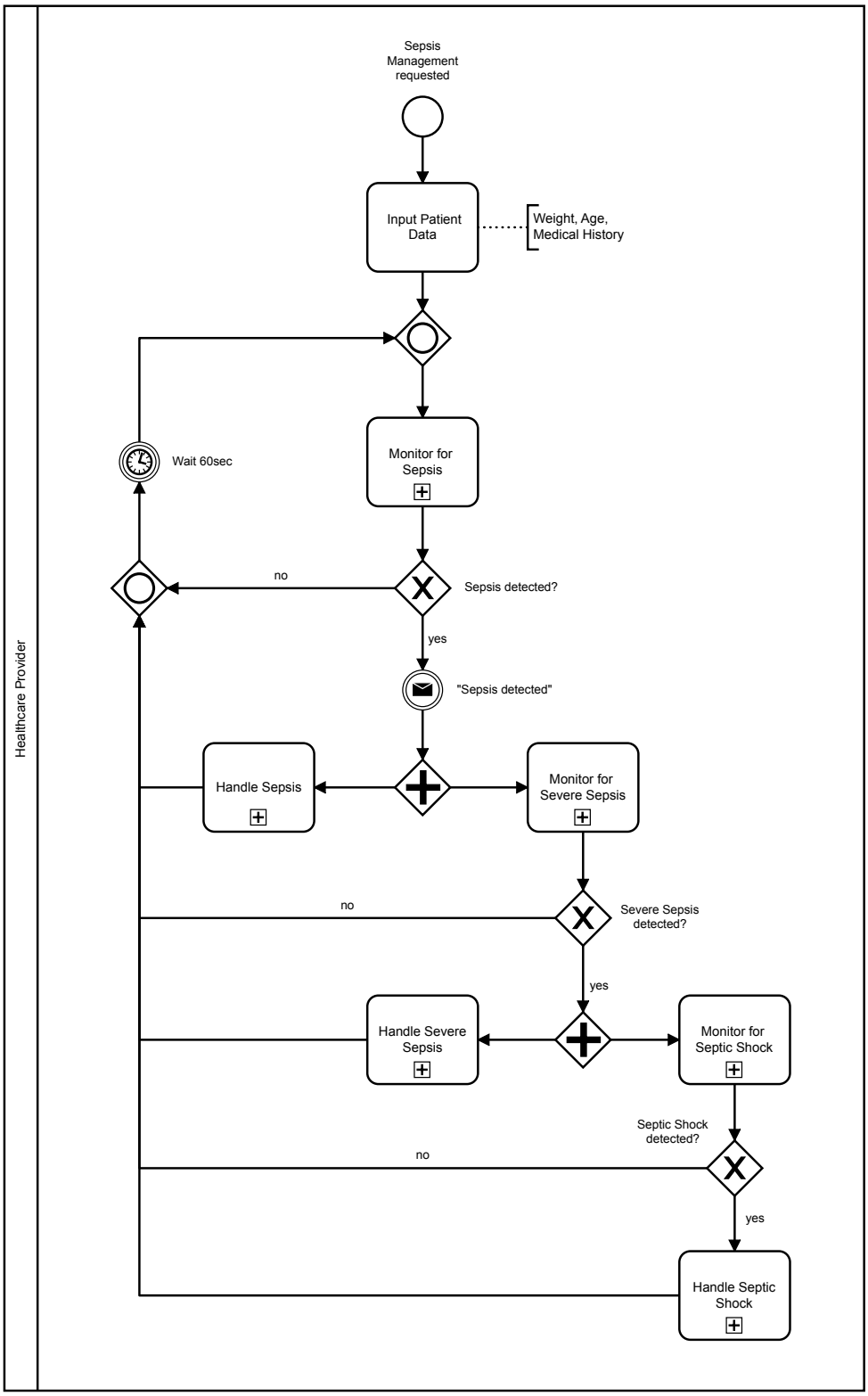

Abbildung 8.2 – *Sepsismanagement in BPMN*

8.3 AUSFÜHRUNG VON PROZESSEN

Ist ein Prozess einmal vollständig in BPMN abgebildet und mit einem BPMS computerisiert, so kann er dort ausgeführt werden und gelangt damit in die zweite Stufe des Prozessmanagement. Vollständig modelliert bedeutet insbesondere, dass auch alle Subprozesse detailliert wurden.

Die meisten BPMS bieten nicht nur eine Komponente für die Prozessausführung an, sondern auch eine Validierungsfunktion, die prüft, ob ein in BPMN modellierter Prozess syntaktisch und semantisch geeignet dafür ist. Für die Ausführung selbst wird in einer Laufzeitumgebung je Prozess (i .e. Kollaborationsdiagramm) eine Instanz angelegt, mit initialen Daten befüllt, gestartet und deren Durchlauf protokolliert. Je nach Anwendungsfall entstehen dabei unterschiedliche Ergebnisse. So soll das sogenannte Workflow-Management als eine Art Tracking-System den reibungslosen Ablauf von Prozessen unterstützen, indem es Fristen überprüft, Nachrichten an Prozessbeteiligte schickt, Eskalationen auslöst etc.

Für ein Prozessmonitoring wiederum werden umfangreiche Darstellungen unterschiedlicher Formate - meist grafisch - erzeugt, die in einem sogenannten Dashboard zusammengefasst, die Analyse des jeweiligen Prozesses ermöglichen.

Ein weiterer, populär gewordener Anwendungsfall für die Prozessausführung ist die Robotic Process Automation (RPA), also Software-Roboter oder „Bots", die dialoglos sich wiederholende und regelbasierte Aufgaben automatisieren [103].

Technisch betrachtet ist es für die Prozessausführung notwendig, aus einem BPMN-Kollaborationsdiagramm ausführbaren Quelltext zu generieren. Dies geschieht für gewöhnlich mit Hilfe der Business Process Execution Language (BPEL) und wird intrinsisch vom BPMS übernommen. BPEL ist ein Industriestandard zur Automatisierung von Geschäftsprozessen auf Basis von Webservices. Da sowohl BPMN als auch BPEL standardisierte XML-Anwendungen sind, sind Austausch, Interoperabilität und Integration gewährleistet. Anzumerken ist, dass die Ausführung von Prozessen in einem BPMS ausschließlich auf dem Zielsystem des BPMS selbst möglich ist, also in der Regel als cloud-basierter Webservice.

8.4 SYNERGIEN, LIMITATIONEN UND TRANSFER

Tatsächlich ergeben sich eine Reihe von Ähnlichkeiten, Parallelen und Synergien zur Methodik, die wir zur Lösung der Problemstellung in dieser Arbeit herangezogen haben. Diese und ein möglicher Transfer und/oder eine mögliche Substitution sind nachfolgend beschrieben.

Zunächst einmal enthalten beide Ansätze - Software-Framework und Business Process Management System - Komponenten für die Modellierung und für die Ausführung von Prozessen. Folglich, theoretisch und generell sollte daher auch die Verarbeitung eines klinisch-therapeutischen Prozesses mit einem konventionellen BPMS möglich sein. Das BPMS entspricht in seiner Gesamtheit dem Software-Framework und die BPMN entspricht der grafischen Modellierungsnotation DiaFlux (Abschnitt 5.2.2). Die in d3web enthaltene Ausführungseinheit ist gleichbedeutend mit dem Laufzeitsystem eines BPMS und seiner Quelltextsprache

BPEL. Dagegen bieten gängige BPMS kein Knowledge Wiki wie KnowWE und somit auch nicht die Möglichkeit einer Single Source of Truth (SSOT). Einer Wissensbasis, die die formal, explizite Modellierung eines klinisch-therapeutischen Prozesses repräsentiert, kommt die Sammlung aller BPMN-Kollaborationsdiagramme inklusive deren Dokumentation gleich.

D3web zeichnet sich durch die Bereitstellung einer Vielzahl von Problemlösungsstrategien (Abschnitt 3.4) aus, die eine hohe Flexibilität hinsichtlich potenziell computerisierbarer klinisch-therapeutischer Prozess sicherstellt. Das trifft vor allem auf das temporale Schließen zu, für das in d3web eigens die Erweiterung Time-DB (Abschnitt 5.2.3) erschaffen wurde. Obwohl der Umfang der BPMN immens ist und denjenigen von DiaFlux bei weitem übersteigt, sind dort keinerlei Modellierungselemente für die Repräsentation temporaler Logiken vorgesehen. Letztlich beschränken sich die Problemlösungsstrategien eines BPMS alleinig auf das regelbasierte Schließen.

Weiterhin limitierend für den Einsatz eines BPMS als Software-Framework könnte die Anbindung an ein spezifisches Zielsystem sein, das ein anderes ist als das des BPMS, besonders wenn eine autonome Steuerung im Closed-Loop-Modus (i. e. Vollautomatik) gewünscht ist. Der Aufwand für „Spezialimplementierungen" kann beträchtlich sein oder das Vorhaben ist gar zum Scheitern verurteilt, wenn die Technologie des Zielsystems dies nicht zulässt. Oder der Hersteller.

Bezüglich des mittels der zwei Prinzipien und drei Paradigmen gesponnenen roten Fadens dieser Arbeit lässt sich folgendes feststellen:

Leitprinzip - Vom Programmieren zum Spezifizieren. Kann mit einem BPMS voll umgesetzt werden.

Design-Prinzip - Trenne invariante von variablen Anteilen. Ist mit einem BPMS intrinsisch realisiert.

Paradigma I - Zwei Freiheitsgrade. Der Freiheitsgrad „multiple Zielsysteme" kann von einem BPMS nur mit obiger Limitation der Anbindung an Zielsysteme erreicht werden, während der Freiheitsgrad „multiple Prozesse" mit der Zweckbestimmung eines BPMS garantiert ist.

Paradigma II - Zweiphasiger Nutzungszyklus. Kann mit einem BPMS voll umgesetzt werden.

Paradigma III - Hybrides Vorgehensmodell. Ist im BPMS nicht vorgesehen.

Trotz einiger Limitationen und Nachteile bringt die Substitution des Software-Frameworks mit einem BPMS doch auch vielerlei Vorteile, wie beispielsweise Kostenersparnis, Effizienz- und Qualitätssteigerung, Zukunftssicherheit und reduzierte technische Risiken durch

- Nutzung weltweiter Industriestandards
- Nutzung weit verbreiteter IT-Systeme inklusive deren umfassende Dokumentation, Schulungsmaterialien etc.
- Sicherstellen technologischer Aktualität
- Nutzung anerkannten Fachwissens aus der Disziplin Prozessmanagement, insb. Methodiken, Systematiken, Konventionen etc.

- Austausch mit einer breiten Benutzergruppe, Fachgremien, Organisationen etc.
- Integration in vorhandene IT-Systeme des Unternehmens

Ein konkretes Beispiel für einen signifikanten Vorteil eines BPMS als Software-Framework sei mit dem sogenannten *Prozess-Mining* gegeben. Prozess-Mining ist eine Technik, die darauf abzielt, Prozesse in einer Organisation durch die Analyse von Ereignisprotokollen besser zu verstehen und zu optimieren. Dabei werden Daten aus IT-Systemen verwendet, um die tatsächlichen Abläufe zu rekonstruieren, zu visualisieren und zu analysieren. Prozess-Mining hilft Unternehmen, Schwachstellen zu identifizieren, Ineffizienzen zu beseitigen und die Leistung der Prozesse zu verbessern. Angewandt auf Entwicklung und Betrieb von wissensbasierten Software-Anwendungen wäre Prozess-Mining von großem Nutzen, vor allem für das wichtige Thema Post-Market Clinical Follow-Up (PMCF), siehe Abschnitt 6.8, oder für die Optimierung von Evaluierungsverfahren.

8.5 FAZIT UND EMPFEHLUNGEN

Auf den ersten Blick mögen die identifizierten Nachteile und Limitationen aus dem Vergleich des Software-Frameworks mit einem konventionellen BPMS ernüchternd wirken und von etwaigen Transfers und/oder Substitutionen abraten. Aus den Erfahrungen, die wir während der Entwicklung des SWFW, aber hauptsächlich aus Entwicklung und Betrieb mehrerer wissensbasierter Software-Anwendungen (siehe Kapitel 7) gemacht haben, lassen sie sich wie folgt bewerten:

Über ein Knowledge Wiki zu verfügen, dass nicht nur die Zusammenarbeit aller Projektbeteiligten gestattet, sondern auch als XDK dient und zudem als universale Dokumentenquelle dient, ist sicherlich ein hoher Komfortfaktor. Andererseits bieten mittlerweile alle modernen BPMS Funktionalitäten an, die eine synchronisierte Kollaboration erlauben. Und auch KI-basierte Chatbots haben bereits Einzug in diese IT-Systeme gehalten.

Kritischer dagegen könnte das Fehlen des temporalen Schließens sein. Insbesondere, da von Klinikern immer wieder auf dessen Wichtigkeit hingewiesen wurde. Doch auch hier hat die Erfahrung gezeigt, dass von der Fülle der mit der Time-DB implementierten Funktionalitäten in der Realität lediglich einige wenige genutzt wurden. Ähnliches gilt für die bei Entwicklungsstart des Software-Frameworks eingeforderte Vielzahl von Problemlösungsstrategien, siehe Abschnitt 3.4. Ausreichend und typisch für praxisnahe klinisch-therapeutische Prozesse ist tatsächlich das regelbasierte Schließen. Bis auf einige akademisch-experimentelle wissensbasierte Software-Anwendungen reichte bisher für sämtliche andere die regelbasierte Problemlösungsstrategie.

Der geringste Aufwand und das geringste Risiko für die Anbindung einer wissensbasierten Software-Anwendung an ein Zielsystem ergibt sich, wenn das entsprechende Zielsystem identisch ist zu dem des BPMS. Je proprietärer das Zielsystem, desto höher fallen Aufwand und Risiken aus. Jedoch entsteht der Aufwand für die Anbindung ohnehin, also unabhängig davon, ob ein Software-Framework oder ein BPMS verwendet wird. Ausschlaggebender sind vielmehr technische Machbarkeit

und Risiko, die beide beim Software-Framework niedriger ausfallen, denn dort haben wir die Quelltexte in eigener Hand.

Die Notwendigkeit, ein hybrides Vorgehensmodell zu erschaffen, ergab sich aus der Tatsache, dass zunächst einmal das Software-Framework entstehen musste - per konventioneller Software-Entwicklung, um danach wissensbasierte Software-Anwendungen entwickeln zu können - per Knowledge Engineering. Der Clinical Knowledge Engineering Process (CKEP) (siehe Abschnitt 4.6) kann und sollte auch mit einem BPMS uneingeschränkt Anwendung finden und könnte sogar integriert und/oder ergänzt werden mit einem BPM-Lebenszyklus, wie er beispielsweise von Dumas et al. [37] vorgestellt wurde.

Subsumierend ergibt sich unmittelbar die Empfehlung, mit folgender Konstellation die genannten Vorteile eines BPMS gegenüber unserem Software-Framework zu nutzen: Software-Anwendungen, die sich mit dem BPMS dasselbe Zielsystem teilen und ausschließlich der Entscheidungsunterstützung (i. e. Open-Loop) dienen. Hierbei entstehen nahezu keinerlei Risiken, der Aufwand für Implementierung und Pflege ist überschaubar und technische Machbarkeit und funktionelle Erweiterbarkeit ist durch das jeweilige BPMS gegeben. Ein Idealfall entsteht, wenn das Zielsystem ein omnipotentes PDMS ist, wie in Abschnitt 7.3 angedeutet.

Alle weiteren Konstellationen, die heterogene Zielsysteme und/oder invasivere Betriebsarten, wie die Halb- oder Vollautomatik, anstreben, bedürfen einer sorgsamen und präzisen Projektplanung bezüglich technischer Machbarkeit, Risiken und Aufwänden.

Wenn ein klares Zielbild für Motivation, Zweckbestimmung und perspektivische Strategie für die jeweilige Software-Anwendung vorliegen, so lohnt es sich in jedem Fall, den Einsatz eines BPMS genauer zu untersuchen.

Eine visionäre Vorstellung solch einer Modernisierung der Technologie ist die Bereitstellung einer standardisierten, qualitätsgesicherten Bibliothek für klinisch-therapeutische Prozesse. Was bedeutet, das diese Prozesse als integraler Bestandteil des unternehmensweiten Geschäftsprozessmanagement betrachtet werden, jedoch zusätzlich auch als Teil des unternehmensweiten Produktportfolios.

KAPITEL 9

DISKUSSION UND AUSBLICK

Die vorliegende Arbeit beschreibt die Konzeptionierung, Entwicklung, Evaluierung, Zulassung und den Betrieb eines generativen Software-Framework für die elektronische Modellierung und Ausführung einer Vielzahl medizinischer Leitlinien in einer Vielzahl von Medizinprodukten.

Das Software-Framework verfolgt das primäre Ziel, zusammen mit den jeweiligen Wissensexperten flexibel und effizient klinisch-therapeutische Prozesse aus medizinischen Leitlinien zu extrahieren, zu spezifizieren und die Wissensinhalte so zu modellieren, dass sie in einem Zielsystem - hier: Medizinprodukt - ausgeführt und als wissensbasierte Software-Anwendung bereit gestellt werden können.

Primäres Ziel der mit dem Software-Framework generierten, wissensbasierten Software-Anwendungen für die Anästhesiologie ist die Optimierung und Standardisierung klinisch-therapeutischer Prozesse zur Steigerung der Diagnostik und Behandlungsqualität, zur Reduktion von Behandlungskosten sowie zur Entlastung des Klinikpersonals. Damit zahlen sie nachweisbar auf die übergeordneten Ziele ein, die mit dem Quadrupel-Aims des Institute for Healthcare Improvement (Abschnitt 1.1) ausgerufen wurden.

Die Problemstellung aus Abschnitt 1.2 birgt einige Herausforderungen, die es zu bewältigen gilt und deren Lösungen gleichsam die zentralen wissenschaftlichen Beiträge darstellen.

Da ist zunächst der Anwendungsbereich Anästhesiologie mit seinen hoch wissensintensiven, hochspezialisierten und komplexen klinischen Prozessen in einem hochtechnologischen und psychisch belastenden Kontext. Das entsprechende medizinische Wissen kann nur vom klinischen Personal stammen, entweder unmittelbar oder mittelbar in wissenschaftlich aufbereiteter Form von beispielsweise medizinischen Leitlinien (Abschnitt 1.2). Gerade das im Leistungsdruck des klinischen Alltags eingebundene Personal ist jedoch kaum, selten und/oder nur kurz verfügbar. Und medizinisches Wissen aus der Fachliteratur ist zuweilen sehr umfangreich und bedarf einer zeitaufwändigen Analyse, um klinische Prozesse zu identifizieren und zu formalisieren.

Mit der Nutzung eines Knowledge Wiki als integrierte Entwicklungsumgebung (Abschnitt 5.2.1) begegnen wir dem Problem der Verfügbarkeit des Klinikpersonals, was die Zusammenarbeit zwischen Ärzten, Pflegepersonal und Software-Entwicklern effizienter macht. Das konnte uns durch Umfragen und Gesprächen mit klinischen und wissenschaftlichen Kooperationspartnern bestätigt werden. Für eine möglichst effiziente Zusammenarbeit nutzen wir Methoden des Knowledge Engineering, im Speziellen ein eigenes, Clinical Knowledge Engineering Process (CKEP) genanntes Vorgehensmodell (Abschnitt 4.6), mit dem auch die Extraktion des Wissens aus medizinischen Leitlinien vereinfacht wird. Das verwendete Knowledge Wiki bietet darüber hinaus noch weitere Vorzüge wie zentrale Dokumentenquelle (Single Source of Truth), Ausführung von Wissensbasen und Testfunktionalitäten. Obendrein ermöglicht es die Computerisierung der meist nicht-elektronisch vorliegenden medizinischen Leitlinien.

Die Kernanforderung der Multiplizität, siehe Abschnitt 1.4, beschert dem Software-Framework zwar die generativen Fähigkeiten und damit Flexibilität und Wiederverwendbarkeit, fordert aber auch ihren Tribut. So können klinisch therapeutische Prozesse äußerst umfangreich und heterogen sein, was ihren Anwendungsbereich, ihre fachliche Komplexität aber auch ihre Präzision und grundsätzliche Eignung betrifft. Gleiches gilt für die designierten Zielsysteme, die eine oder mehrere wissensbasierte Systeme in der Anästhesiologie betreiben sollen. Je proprietärer und heterogener deren Technologie ist, desto höher der Implementierungsaufwand und das Risiko der technischen Machbarkeit. Dies intensiviert sich noch durch die drei zu realisierenden Betriebsarten Entscheidungsunterstützung, Halbautomatik und Vollautomatik (Abschnitt 5.5).

Gelöst haben wir die Problematik der Multiplizität über Konzeptionierung, Entwurf und Implementierung einer flexiblen Systemarchitektur, die das zentrale Design-Prinzip „trenne invariante von variablen Anteilen" konsequent umsetzt (Abschnitt 4.3) und damit dem Paradigma *Zwei Freiheitsgrade* Rechnung trägt (Abschnitt 4.4). Entwurfsmuster innerhalb der Systemarchitektur sorgen dafür, dass möglichst viele Komponenten invariant bleiben, variable dagegen möglichst einfach und isoliert modifizierbar sind. Außerdem lassen sich weitere fachliche Komponenten unkompliziert ergänzen. Eine dieser fachlichen Komponenten ist für die technische Anbindung an ein Zielsystem zuständig - Freiheitsgrad 1, während eine weitere die den jeweiligen klinischen Prozess computerisierte Wissensbasis repräsentiert - Freiheitsgrad 2. Von diesen beiden essenziellen Komponenten kann es theoretisch beliebig viele geben, sodass eine maximale Multiplizität gewährleistet ist.

Eine weitere Herausforderung besteht in der Umsetzung des Leitprinzips „vom Programmieren zum Spezifizieren", womit gemeint ist, möglichst viel Quelltext zu generieren und das manuelle Programmieren zu minimieren. Das schließt auch den komfortablen Transfer der Quelltexte zum jeweiligen Zielsystem mit ein. Idealerweise erfolgt das Spezifizieren zusammen mit dem oder den Wissensexperten in einer interaktiven Art und Weise.

Auch hier zahlt sich die Verwendung eines Knowledge Wiki aus, vor allem, da es auch für weniger technikaffine Wissensexperten die Prozessmodellierung mittels einer grafische Notation vereinfacht. Ferner haben wir ein weiteres, als *zweiphasi-*

ger Nutzungszyklus bezeichnetes Paradigma eingeführt, das mit den sequentiellen Phasen Modellierung und Ausführung via Definition von Rollen, Aktivitäten und Konventionen die Spezifikation wissensbasierter Software-Anwendungen regelt und vereinfacht (Abschnitt 4.5).

Wissen im Allgemeinen und medizinisches Wissen im Besonderen ändert sich häufig. Folglich müssen auch Software-Anwendungen, die umfangreich solch hochspezialisiertes Wissen einbetten, mit diesem Fakt umgehen können. Dazu kommt, dass dieses Phänomen nicht nur die wissensbasierten Systeme in der Anästhesiologie als solche betrifft, sondern ebenfalls Rahmenbedingungen wie regulatorische und legislative Auflagen, die erfahrungsgemäß gleichfalls ständigen Anpassungen, Ergänzungen und Neuerungen unterworfen sind. Akutes Beispiel ist die regulatorische Forderung aus der Medical Device Regulation zu einer klinischen Nachbeobachtung (engl. Post-Market Clinical Follow-Up), siehe Abschnitt 6.8.

Wie kann also auf derlei Änderungsanforderungen effizient und zuverlässig reagiert werden? Die hier vorgestellten drei Paradigmen beantworten auch diese Frage, zusammen mit Leit- und Design-Prinzip. Es gilt: je größer das Ausmaß invarianter Anteile, je klarer und umfassender die Spezifikation der Wissensinhalte per Vorgehensmodell geregelt ist und je flexibler sich die Systemarchitektur zeigt, desto geringer fallen Aufwand und Risiko für Änderungen aus.

Schließlich stellt auch die überaus wichtige Thematik der Evaluierung - also Verifikation, Validierung und klinische Evaluierung von Software-Framework und besonders der generierten wissensbasierten Systeme in der Anästhesiologie - eine anspruchsvolle Herausforderung dar. Die erforderlichen qualitätssichernden Maßnahmen für so hochregulierte Medizinprodukte können sehr schnell vertretbare Aufwendungen bezüglich Kosten, Zeit sowie wirtschaftliches und technisches Risiko übersteigen.

Prinzipiell sind diese Aufwendungen beim Software-Framework deutlich geringer, als bei dessen Erzeugnissen, siehe Abschnitt 6.2. Deren Evaluierung mündet meistens in einer kosten- und zeitintensiven klinischen Studie. Bis es so weit kommt, haben sich von den gängigen Evaluierungsverfahren (Abbildung 6.1, Seite 82), besonders die Software-interne Validierung und der Einsatz von Software-Simulationen bewährt und etabliert. Begleitend zu dieser Arbeit wurde eigens dafür eine Software-Simulation entwickelt und in Betrieb genommen, die ähnlich flexibel und erweiterbar ist wie das Software-Framework selbst, siehe Abschnitt 6.5.

Resümierend aus den Erkenntnissen, Ergebnissen und den Erfahrungen aus Entwicklung und Einsatz des Software-Frameworks sowie den damit hervorgebrachten wissensbasierten Anwendungen für die Anästhesiologie, lässt sich feststellen:

Das Software-Framework hat die Zweckbestimmung sowie die damit verbundenen Anwendungsfälle (Abschnitt 4.2) erfüllt und vollumfänglich implementiert, was durch entsprechende Evaluierungen und Kundenrückmeldungen bestätigt wurde. Man kann ergo von einer gegebenen Praktikabilität und Betriebsbewährtheit sprechen. Es konnten zahlreiche wissensbasierte Systeme für die Anästhesiologie entwickelt werden (Kapitel 7), die den angestrebten klinischen Mehrwert erzielten, was mittels klinischer Studien und anderer wissenschaftlicher Untersuchungen

nachgewiesen wurde, siehe [3], [53], [90], [17], [15], [73] und deren Zusammenfassungen in Abschnitt 7.2, Abschnitt 7.5, Abschnitt 7.4 und Abschnitt 7.6. Damit sind gleichsam auch die übergeordneten Zielsetzungen des Quadruple-Aims (Seite 2) sowie die generellen Vorteile medizinischer Leitlinien (Seite 4) quantitativ nachweisbar erreicht worden.

Andererseits liegen uns wenig quantitative Fakten zur Beurteilung der Effizienz unseres Software-Frameworks vor. Hier ist es wünschenswert, dies über die Nutzung von Metriken und vergleichenden Betrachtungen messbar zu machen. Auch bietet sich der Einsatz von Reifegradmodellen wie SPICE (Software Process Improvement and Capability Determination) oder CMN (Capability Maturity Model) dafür an.

Das Design-Prinzip der Trennung invarianter von variablen Anteilen gewährleistet eine hohe Wiederverwendung von mindestens der drei methodischen Paradigmen und der Systemarchitektur, unabhängig von der Technologie der Implementierung.

Im Sinne eines „Lessons learned" mussten wir allerdings auch schmerzliche Erfahrungen machen. Die zu Tage gekommenen kritischen Pfade leiten wir jeweils über Kernfragen ein.

Wie kommen wir zu geeignetem medizinischen Wissen, das zum einen präzise und kompakt genug ist, aber zum anderen eine hinreichende, klinische Evidenz aufweist?

Diese Frage adressiert die inhärente Diskrepanz zwischen Präzision und praktischer Tauglichkeit medizinischen Wissens auf der einen und der für die Produktzulassung und Vermarktung unerlässlichen klinischen Evidenz auf der anderen Seite.

In einem hypothetischen Extremszenario liefert ein einzelner Arzt das medizinische Wissen für die Computerisierung eines klinisch-therapeutischen Prozesses, basierend auf seinen fundierten, praktischen Erfahrungen, lokal aus seiner Klinik. Dieses Wissen mag präzise und kompakt, ja sogar konkret, spezifisch sein bis hin zu einzelnen Medizingeräten, doch ist es vermutlich wenig strukturiert und nicht nachweisbar evidenzbasiert. Gemäß der S-Klassifikation der evidenzbasierten Medizin erreicht es maximal den Wert S1, siehe Abschnitt 2.1.3.

Das bedeutet, dass solch ein Design-Input zwar einfach und zügig in eine wissensbasierte Anwendung zu implementieren ist, die notwendige Bereitstellung klinischer Daten (Abschnitt 6.7) allerdings langjährige, kostspielige Evaluierungen erfordern wird, um eine Produktzulassung zu erwirken. Auch ist in diesem Fall die subjektive Akzeptanz der Anwender eher gering, was einer erfolgreichen Nutzung entgegensteht.

Im anderen Extremszenario wird eine S3-Leitlinie (Abschnitt 2.1.3) als Design-Input herangezogen. Also eine medizinische Leitlinie mit der höchsten klinischen Evidenz, wie wir sie inhaltlich in Tabelle 2.2, Seite 17 skizziert haben. In einem solchen Fall sind klinische Daten meist schon vorliegend und eine Produktzulassung daher zeitnah zu machen. Konträr dazu sind die Inhalte einer S3-Leitlinie jedoch eher allgemein gehalten und bisweilen überbordend - soll sie doch ein möglichst breites Spektrum abbilden - und der Aufwand im Knowledge Engineering, daraus

eine Prozessmodellierung zu erzeugen, ist erheblich bis inakzeptabel. Um weder in das eine noch in das andere Szenario zu geraten, ist es unerlässlich, diejenige Balance zu finden, mit der ein Entwicklungsprojekt akzeptabel realisiert werden und gleichzeitig Akzeptanz, Anwendung und Erfolg im klinischen Einsatz erzielen kann.

Wie können wir möglichst viele Zielsysteme unterschiedlichster Technologie erreichen, ohne dass der Projekterfolg gefährdet wird?

Mit der zweiten Frage thematisieren wir das kritische Thema der technischen Anbindung wissensbasierter Systeme. Das klang schon weiter oben beim Thema Multiplizität an. Hier liegt der Fokus auf Implementierungsaufwand und dem Risiko der technischen Machbarkeit. Gerade wenn für sehr heterogene, proprietäre, lang existierende Zielsysteme eine Software-Anwendung nachträglich entwickelt werden soll, kann sich mitunter erst spät herausstellen, dass es technisch gar nicht möglich ist, das Zielsystem anzubinden. Dann sind aber bereits eine Menge Zeit verstrichen und hohe Entwicklungskosten entstanden. Das Risiko liegt besonders hoch, wenn eine autonome Steuerung des Zielsystems angestrebt ist.

Die klare Empfehlung lautet hier, die technische Machbarkeit in jedem Falle sorgfältig zu prüfen und eine bedächtige, eher risikoscheue Herangehensweise zu bevorzugen. Das ideale Zielsystem ist mit dem Laufzeitsystem des Software-Frameworks identisch und die wissensbasierte Anwendung wird nicht a posteriori sondern simultan mit selbigem entwickelt.

Wie können wir ein Erzeugnis des Software-Frameworks ausreichend evaluieren, ohne dass das Entwicklungsprojekt wirtschaftlich untragbar wird?

Die dritte Frage schließlich beschäftigt sich mit dem heiklen Thema der Evaluierung einer wissensbasierten Software-Anwendung. Auch hier gilt es, die richtige Balance zu finden zwischen der minimalen Erfüllung zwingend notwendiger Evaluierungen, einem unternehmenseigenen Qualitätsanspruch und den sie erzeugenden monetären und zeitlichen Aufwand. Wir haben Entwicklungsprojekte erlebt, die kurz vor einem Inverkehrbringen standen, dann aber wegen Unwirtschaftlichkeit der klinischen Evaluierung eingestellt werden mussten.

Es ist nicht von der Hand zu weisen, dass die Technologien, nicht aber die Methodik, mit denen wir das Software-Framework entwickelt haben und gegenwärtig immer noch betreiben, überwiegend proprietär, wenig verbreitet und mittlerweile recht betagt sind. Das ist auch der Grund, warum wir in Kapitel 8 untersucht haben, inwieweit man diesem Umstand mit Anleihen aus dem Bereich des Geschäftsprozessmanagement Rechnung tragen kann. Dabei haben sich durchaus attraktive Möglichkeiten eröffnet, wie in Abschnitt 8.4 beschrieben, von denen exponiert der Ersatz des Software-Frameworks durch ein konventionelles Business Process Management System (BPMS) erwähnt sei.

Greifen wir die im Resümee beleuchteten Schwachstellen bezüglich der verwendeten Technologien auf, so lassen sich drei interessante Ansätze identifizierten, mit denen sich diesen zukünftig begegnen lässt.

Als Zwischenschritt auf dem Weg zum Ersatz des Software-Frameworks könnte es lohnenswert sein, die im Knowledge Wiki verwendete, proprietäre grafische Notation zur Modellierung von Prozessen, mit dem weltweiten Industriestandard der Business Process Model and Notation (BPMN) zu ersetzen. Mächtigkeit, Transparenz und Verbreitung dieser alternativen grafischen Notation könnten Zukunftssicherheit, Interoperabilität und Portierbarkeit erhöhen. Dazu müsste das frei verfügbare Knowledge Wiki entsprechend erweitert und angepasst werden, was insbesondere für die Zeitdatenbank (Abschnitt 5.2.3) gilt, die jedoch gegebenenfalls mit BPMN-Funktionalitäten nach implementiert werden kann oder aber einfach erhalten wird.

Weiterhin attraktiv ist die Idee, auch für die autonome Steuerung von Zielsystemen mittlerweile hervorgebrachte Standards zu nutzen. Zu diesem Zwecke bietet sich die IEEE P11073-10107 *Standard for Nomenclature for External Control of Medical Devices* an, die Definitionen, Nomenklatur, Kommunikationsprotokolle und Kommandos für die externe Steuerung von Medizingeräten liefert [120]. Sie erweitert die IEEE 11073 Familie von Standards, die sich auf die Interoperabilität von medizinischen Geräten und Systemen konzentrieren und unter dem Titel *Service-oriented Device Connectivity (SDC)* geführt werden, siehe dazu auch den Anwendungsbericht von Andersen [56]. Diese Standards wurden entwickelt, um den Austausch von Informationen zwischen medizinischen Geräten und IT-Systemen im Gesundheitswesen zu erleichtern und zu standardisieren. Technologisch dürfte es ein Leichtes sein, etwaige mit diesem Standard konforme Medizinprodukte autonom zu steuern, was ebenfalls der Zukunftssicherheit und der Integrationsfähigkeit zu Gute kommt.

Im Grunde genommen ähnelt unser Software-Framework in vielerlei Hinsicht den derzeit populären No-code/Low-Code Plattformen (LCNC) [96]. Sie stellen quasi eine weitreichende Fortführung der Kernideen des Software-Frameworks in die Zukunft dar und lohnen daher sicherlich der näheren Betrachtung.

Bezogen auf den hier vorgestellten Entwicklungsprozess und das dazugehörige hybride Vorgehensmodell (Abschnitt 4.6) können ebenfalls zukunftsweisende Perspektiven aufgezeigt werden.

So hat seit geraumer Zeit das agile Manifest Einzug in das Geschäftsprozessmanagement gehalten und das agile BPM begründet. Badakhshan et al. haben 2019 eine sehr umfangreiche systematische Literaturrecherche veröffentlicht [88]. Agiles Engineering, so wie wir es aus der Software-Entwicklung kennen, auf die Entwicklung und Pflege wissensbasierter Anwendungen für die Anästhesiologie übertragen, kann beispielsweise eine kontinuierliche Prozesskonformität (engl. Continuous Process Compliance) systemisch zusichern. Praktisch müsste dazu das agile BPM mit dem Clinical Knowledge Engineering Process (CKEP) kombiniert und etabliert werden.

Ähnlich wie bei den technologischen Zukunftstrends finden wir zu Methodik und Vorgehensmodell mit der KI-gestützten Software-Entwicklung einen hochinteressanten Weg, um u. a. unser Design-Prinzip zu perfektionieren. Das Fraunhofer-Institut für Experimentelles Software Engineering IESE [116] beschäftigt sich intensiv mit diesem Thema und bietet eine Menge hilfreicher Artikel an, von de-

nen ein Beitrag über generative KI im Software Engineering besonders lesenswert ist [115].

Ein weiterer Ausblick betrachtet den Anwendungs- und Einsatzbereich des Software-Frameworks hinsichtlich des Zielsystems. Bei der Vorstellung der wissensbasierten Software-Anwendung für das Erkennen und Managen septischer Erkrankungen (Abschnitt 7.3) haben wir bereits die Idee geäußert, ein Patientendaten-Managementsystem (PDMS) als universelles Zielsystem einzusetzen. Hintergrund war die einfachere technische Implementierung, da PDMS in vielen Fällen auf Computern mit Standard-Betriebssystemen lauffähig sind. Hier liegt der Fokus allerdings mehr auf der Ausweitung des Einsatzbereiches, denn ein PDMS als zentrale Datenbank in der Anästhesiologie kann weitaus mehr klinische Daten bereitstellen und somit weitaus mehr klinisch-therapeutische Prozesse versorgen als einzelne Medizingeräte. Der nächste naheliegende Schritt wäre dann eine weitere Ausweitung vom PDMS auf das globale Krankenhausinformationssystem.

Abschließend stellt sich die naheliegende Frage, ob das Software-Framework auch für die Modellierung und Ausführung von Prozessen in nicht-medizinischen Domänen geeignet ist. Diese Frage lässt sich eindeutig bejahen. Erforderlich ist hierfür lediglich die Identifikation geeigneter Prozesse in den jeweiligen fachfremden Anwendungsbereichen.

Die beeindruckende Vielzahl unterschiedlicher Blicke in die Zukunft überrascht dann doch und sichert wohl für einen beachtlichen Zeitraum eine reichhaltige Beschäftigung mit dem spannenden Thema „Von der Leitlinie zur Anwendung: Ein Framework zur software-gestützten Umsetzung klinischer Prozesse".

So bleibt nur noch zu hoffen und wünschen, dass die Ergebnisse und Erkenntnisse dieser Arbeit ein wenig dazu beitragen können, das nachhaltig inspirierende Motto von Robert „Bob" Kacmarek (1949 - 2021) [94] in die Tat umzusetzen. Der nämlich hat uns Ingenieuren 2001 während einer Pro-Con-Debatte auf dem International Symposium on Intensive Care & Emergency Medicine in Brüssel zugerufen:

Freeing the Clinician for the Art of Medicine

* * * * *

ABKÜRZUNGSVERZEICHNIS

AWMF
Arbeitsgemeinschaft der Wissenschaftlichen Medizinischen Fachgesellschaften. 3, 16

BDZT
Befund-Diagnose-Ziel-Therapie. 21, 22, 30, 33, 34, 43, 95

BMBF
Bundesministerium für Bildung und Forschung. 102

BPEL
Business Process Execution Language. 112, 113

BPM
Business Process Management. 106, 122, 149

BPMN
Business Process Model and Notation. 107, 108, 110, 112, 113, 122

BPMS
Business Process Management System. 106, 108, 112–115, 121

CG
Clinical Guideline. 3, 18, 92, 99, 103

CKEP
Clinical Knowledge Engineering Process. 53, 55, 103, 115, 118, 122

DMN
Decision Model and Notation. 107, 108

E2E
Ende-zu-Ende. 106

EABPM
European Association of Business Process Management. 106

EBM
evidenzbasierte Medizin. 15, 16, 18

GLIF
Guideline Interchange Format. 5, 6, 40

GUI
Graphical User Interface. 24, 31, 43, 44, 46, 58, 71, 74, 80, 83, 85

HPS
Human Patient Simulator. 85, 86, 89, 101

IHI
Institute for Healthcare Improvement. 2, 117, 133

IKT
Informations- und Kommunikationstechnologien. 102

IMS
Integriertes Managementsystem. 79

ITS
Intensivstation. 95, 102

J2SE
Java 2 Platform, Standard Edition. 58

JNI
Java Native Interface. 59, 74–76, 99

JVM
Java Virtual Machine. 74, 99

KI
künstliche Intelligenz. 28, 114, 122, 123

KIS
Krankenhausinformationssystem. 123

KTP
klinisch-therapeutischer Prozess. 22, 31, 41, 43, 63, 79, 92, 93, 113

LCNC
Low-Code/No-Code. 40, 122

MDCG
Medical Device Coordination Group. 87

MDR
Medical Device Regulation. 87–89, 119

MLM
Medical Logic Module. 5, 6

MPDG
Medizinprodukterecht-Durchführungsgesetz. 88

OWL
Web Ontology Language. 59

PAP
Programmablaufplan. 108, 110

PDCA
Plan-Do-Check-Act. 21, 22

PDMS
Patientendaten-Managementsystem. 14, 23, 43, 47, 92, 95, 96, 102–104, 115, 123

PMCF
Post-Market Clinical Follow-Up. 89, 114, 119

QMS
Qualitätsmanagementsystem. 78, 79

RAWA
Read-All-Write-All. 74

RAWS
Read-All-Write-Some. 74

RPA
Robotic Process Automation. 112

SDR
Software Development Kit. 59

SOP
Standard Operating Procedure. 18

SSOT
Single Source of Truth. 113

SWFW
Software-Framework. 7–12, 14, 15, 20, 22–24, 27, 30, 31, 39–41, 44, 46–49, 51, 52, 55–59, 67–70, 74, 77–79, 81–83, 88, 91–93, 104–106, 112–115, 117–119, 121–123, 133, 135

TMS
Truth Maintenance System. 62

TPL
Textual Programming Language. 40

UML
Unified Modelling Language. 6, 40, 42

UNCOL
Universal Computer-oriented Language. 40

VPL
Visual Programming Language. 40

WB

Wissensbasis. 30, 35, 43, 46, 54, 55, 68, 70, 71, 74, 79, 113, 118

WBSA

wissensbasierte Software-Anwendung. 41, 43, 44, 47, 48, 50, 54, 55, 68, 70, 71, 73, 74, 78–80, 86, 88, 89, 92, 95, 97, 100–103, 114, 115, 133

XDK

Expert System Development Kit. 50, 51, 54, 59, 66, 67, 114

XPS

Expertensystem. 29–32, 46, 47, 49

GLOSSAR

Abstrakte Fabrik

Entwurfsmuster, das verwendet wird, um eine Gruppe von verwandten oder voneinander abhängigen Objekten zu erzeugen, ohne deren konkrete Klassen explizit spezifizieren zu müssen. 71

Ausführungseinheit

Komponente oder Element eines Software-Systems, das eigenständig ausführbar ist und spezifische Aufgaben innerhalb eines größeren Systems erfüllt. 49, 50, 67, 112

Autorensystem

Software-Umgebung, die Experten oder Entwicklern hilft, regelbasierte oder wissensbasierte Systeme zu erstellen, ohne tiefgehende Programmierkenntnisse zu benötigen. 5–7, 50, 59

Bayessche Netzwerke

Bayessche Netzwerke sind grafische Modelle, die zur Darstellung und Analyse von Wahrscheinlichkeitsbeziehungen zwischen einer Menge von Variablen verwendet werden und die eine Möglichkeit bieten, Unsicherheiten zu modellieren und Wahrscheinlichkeiten abzuleiten. 35

Beatmungsform

Muster, das beschreibt, nach welchem Parameter (Druck, Volumen, Fluss) ein Beatmungsgerät die Atemarbeit steuert und ob und wie der Patient mit dem Gerät interagieren kann. 99, 101, 102

Dashboard

Ein Dashboard ist eine übersichtliche Benutzeroberfläche, die wichtige Informationen, Kennzahlen und Daten aus verschiedenen Quellen verdichtet und grafisch aufbereitet darstellt. 112

Domänenexperte

Experte, Mensch mit ausgewiesener Fachexpertise für einen ausgewählten Wissensbereich (Domäne). 29, 30, 32, 38, 52, 54, 61, 63

Entwurfsmuster

Ein Entwurfsmuster ist eine bewährte, wiederverwendbare Lösung für ein häufig auftretendes Problem in der Software-Entwicklung. 45, 46

I18N

Steht für „Internationalisierung" und bezieht sich auf den Prozess, Software, Produkte oder Inhalte so zu gestalten, dass sie leicht an verschiedene Sprachen und kulturelle Kontexte angepasst werden können. 67

Inverkehrbringen

Erstmalige Bereitstellung eines (Medizin)Produktes auf dem Unionsmarkt Medical Device Regulation, mit Ausnahme von Prüfprodukten. 51, 121

Lastenheft

Das Lastenheft beschreibt die Anforderungen an ein Produkt bzw. an die Projektergebnisse aus Sicht des Auftraggebers. 41, 54, 80, 81, 103, 110

Mortalität

Anzahl der Todesfälle in einer bestimmten Gruppe von Menschen innerhalb eines bestimmten Zeitraums. 95

Ontologie

Beschreibung (Taxonomie) eines Wissensbereiches inklusive der Beziehungen und Ableitungsregeln zwischen den dort verwendeten Begriffen mit Hilfe einer standardisierenden Terminologie. 36, 38, 68

Pflichtenheft

Dokument, das die Anforderungen und Spezifikationen für ein zu entwickelndes Produkt oder System detailliert beschreibt. Es dient als verbindliche Grundlage für die Zusammenarbeit zwischen dem Auftraggeber und dem Auftragnehmer im Rahmen eines Projektes. 49, 81

Problemlösungstyp

Ausgewählter Wissensbereich, innerhalb dessen ein etablierter Lösungsweg für Probleme typisiert werden kann. 30, 33, 34

Programmablaufplan

Grafische Darstellung, die die logische Reihenfolge von Verarbeitungsschritten, Entscheidungen und Abläufen in einem Computerprogramm oder Prozess darstellt. 22–25, 108, 110, 133

Single Source of Truth

Konzept im Datenmanagement, das darauf abzielt, sicherzustellen, dass alle Daten und Informationen innerhalb einer Organisation bzw. eines Anwendungsbereiches aus einer einzigen, zentralen und verlässlichen Quelle stammen. 61, 79, 113, 118

Singleton

Entwurfsmuster, das sicherstellt, dass eine Klasse genau eine Instanz hat, bietet einen globalen Zugriffspunkt auf diese Instanz. 71

Truth Maintenance System

System, das die Konsistenz von Informationen innerhalb eines wissensbasierten Systems gewährleistet, indem es Überzeugungen und deren Rechtfertigun-

gen untersucht, um sicherzustellen, dass alle abgeleiteten Schlussfolgerungen konsistent sind. 36, 62

Vorgehensmodell

Ein systematischer Ansatz oder eine strukturierte Methode, die beschreibt, wie bestimmte Prozesse oder Projekte durchgeführt werden sollen. Es bietet eine klare Anleitung für die Planung, Durchführung und Überwachung von Aufgaben, um definierte Ziele zu erreichen. 51–53, 118, 119, 122

Zielsystem

IT-System, auf dem eine bestimmte Software-Anwendung ausgeführt werden soll. Das können herkömmliche Computer sein, aber auch eingebettete Systeme wie beispielsweise Konsumgüter, Medizingeräte, Fahrzeugtechnik o.ä. 3, 22, 24, 41, 43–46, 48, 55, 67, 68, 71, 79, 83, 92, 95, 99, 104, 112, 117, 118, 122

Zweckbestimmung

Verwendung, für die ein Produkt entsprechend den Angaben des Herstellers auf der Kennzeichnung, in der Gebrauchsanweisung oder dem Werbe- oder Verkaufsmaterial bzw. den Werbe- oder Verkaufsangaben vorgesehen ist. 41, 53, 70, 79, 80, 83, 93, 94, 97, 115, 119

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

PUBLIKATIONSVERZEICHNIS

ARTIKEL AUS FACHZEITSCHRIFTEN

[1] D. Schädler, S. Mersmann, I. Frerichs, G. Elke, T. Semmel-Griebeler, O. Noll, S. Pulletz, G. Zick, M. David, W. Heinrichs, J. Scholz, N. Weiler. „A knowledge- and model-based system for automated weaning from mechanical ventilation: technical description and first clinical application“. In: *Journal of Clinical Monitoring and Computing* Bd. 28, Nr. 5 (2014), S. 497–498. DOI: 10.1007/s10877-013-9489-7.

[2] D. Schädler, S. Mersmann, N. Weiler. „Systematische Evaluierung von SmartCare/BIPAP mit Hilfe eines full-scale Simulators“. In: *Dräger, Lübeck, intern* (2010).

[3] F. Lellouche, J. Mancebo, P. Jolliet, J. Roeseler, F. Schortgen, M. Dojat, B. Cabello, L. Bouadma, P. Rodriguez, S. Maggiore, M. Reynaert, S. Mersmann, L. Brochard. „A Multicenter Randomized Trial of Computer-Driven Protocolized Weaning from Mechanical Ventilation“. In: *Am J Respir Crit Care Med* Bd. 174 (2006), S. 894–900.

[4] S. Leonhardt, S. Mersmann. „Automatisierungstechnik für die künstliche Beatmung – eine Standortbestimmung“. In: *at – Automatisierungstechnik* Bd. 55, Nr. 5 (2007), S. 244–254.

[5] S. Mersmann. „Guideline to Clinical Knowledge Engineering“. In: *Dräger, Lübeck, intern* (2011).

[6] S. Mersmann. „SmartCare® Applications“. In: *Journal für Anästhesie und Intensivbehandlung* (2013), S. 108–111.

[7] S. Mersmann. „SmartCare®: Automatisierung klinischer Leitlinien“. In: *Biomedizinische Technik* Bd. 54 (2009), S. 283–288.

[8] S. Mersmann, K. Kück. „SmartCare™ - Optimizing Workflow Processes in Critical Care through Automation“. In: *Journal of Clinical Monitoring and Computing* Bd. 20, Nr. 2 (2006), S. 119–120.

[9] S. Mersmann, L. Kilian, C. Hörmann. „Kurzbeschreibung einer klinischen Leitlinie für die Beatmungstherapie mit PPS“. In: *Dräger, Lübeck, intern* (2011).

[10] T. Baier-Löwenstein, S. Mersmann. „Modellbeschreibung des SmartPatient v1.1.0“. In: *Dräger, Lübeck, intern* (2012).

BUCHBEITRÄGE

[11] H.-J. Kohl, S. Mersmann. „Künstliche Beatmung“. In: *Medizintechnische Systeme: Physiologische Grundlagen, Gerätetechnik und automatisierte Therapieführung*. Hrsg. von S. Leonhardt, M. Walter. Springer-Verlag, 2016. Kap. 6, S. 145–195. DOI: 10.1007/978-3-642-41239-4_6.

[12] P. Jouvet, S. Mersmann, M. Wysocki. „Pediatric and Neonatal Mechanical Ventilation“. In: Hrsg. von P.C. Rimensberger. Berlin Heidelberg: Springer-Verlag, 2015. Kap. Automation of Weaning in Children, S. 1455–1465. DOI: 10.1007/978-3-642-01219-8_61.

[13] P. Jouvet, S. Mersmann, M. Wysocki. „Textbook on Pediatric and Neonatal Mechanical Ventilation“. In: Hrsg. von P. Rimensberger. New York: Springer, 2012. Kap. Automation of weaning in children.

KONFERENZBEITRÄGE

[14] D. Schädler, S. Mersmann, T. Semmel-Griebeler, O. Noll, W. Heinrichs, N. Weiler. „A knowledge- and model-based system for automated weaning from mechanical ventilation - Technology“. In: *22nd Annual Meeting of the European Society for Computing and Technology in Anaesthesia and Intensive Care (ESCTAIC)*. 2011.

[15] D. Schädler, T. Semmel-Griebeler, O. Noll, W. Heinrichs, S. Mersmann, N. Weiler. „A knowledge- and model-based system for automated weaning from mechanical ventilation - Clinical Application“. In: *22nd Annual Meeting of the European Society for Computing and Technology in Anaesthesia and Intensive Care (ESCTAIC)*. 2011.

[16] G. Zeplin, A.K. Kock, M. Poelker, K. Seidl, S. Mersmann, J. Ingenerf. „Semantische HL7v3 Annotation klinischer Messwerte aus einem PDMS mittels ISO/IEEE 11073 Gerätebeschreibungen zur Anbindung von Smart Applications“. In: *GMDS Jahrestagung*. Mainz, 2011.

[17] K. Saihi, J.C.M. Richard, X. Gonin, T. Krüger, S. Mersmann, M. Dojat, L. Brochard. „Clinical Evaluation of an Automated Open Loop Controller of Inspired Oxygen Concentration for Mechanical Ventilation“. In: *26th ESICM Annual Congress*. 2013.

[18] R. Hatko, D. Schädler, S. Mersmann, J. Baumeister, N. Weiler, F. Puppe. „Implementing an automated ventilation guideline using the semantic wiki KnowWE“. In: *Lecture Notes in Computer Science*. Bd. 7603. EKAW. Springer, 2012, S. 363–372.

[19] R. Hatko, J. Baumeister, G. Meinke, S. Mersmann, F. Puppe. „Anomaly detection in DiaFlux models“. In: *7th Workshop on Knowledge Engineering and Software Engineering (KESE7)*. Bd. 805. San Cristobal de La Laguna, Spain, 2011.

[20] R. Hatko, S. Mersmann, F. Puppe. „Behaviour-Driven Development for Computer-Interpretable Clinical Guidelines“. In: *10th Workshop on Knowledge Engineering and Software Engineering (KESE2014)*. 2014.

[21] S. Mersmann, M. Dojat. „A Flexible System Architecture for Implementation of Protocol-Based Controllers in Mechanical Ventilation“. In: *Proceedings of Workshop "Computers in Anaesthesia and Intensive Care: Knowledge-Based Information Management"*. AIME. Cascais, Portugal, 2001.

[22] S. Mersmann, M. Dojat. „SmartCare™ - Automated Clinical Guidelines in Critical Care“. In: *16th European Conference on Artificial Intelligence*. Valencia, Spain: IOS-Press, 2004, S. 745–749.

PATENTE

[23] S. Mersmann, P. Jolliet, T. Katschewitz, A. Neumann. „Beatmungssystem“. Pat. DE102012003115 (Lübeck). Aug. 2013.

[24] S. Mersmann, W. Buschke, C. Hörmann. „Anästhesiebeatmungsvorrichtung zur automatisierten Beatmung eines Patienten“. Pat. DE102015015440 (Lübeck). Juni 2017.

[25] S. Mersmann, W. Buschke, C. Hörmann. „Anästhesiebeatmungsvorrichtung zur automatisierten Beatmung sowie zur Detektion eines Betriebszustandes hinsichtlich der automatisierten Beatmung“. Pat. DE102015015441 (Lübeck). Juni 2017.

[26] S. Mersmann, W. Buschke, C. Hörmann. „Beatmungsvorrichtung und Verfahren zur automatisierten Beatmung eines Patienten“. Pat. DE102015015439 (Lübeck). Juni 2017.

LITERATURVERZEICHNIS

ABSCHLUSSARBEITEN UND FACHBÜCHER

[27] E. Gamma, R. Helm, R. Johnson et al. *Design Patterns. Elements of Reusable Object-Oriented Software*. Prentice Hall, 1994. ISBN: 978-0201633610.

[28] European Association of Business Process Management, Hrsg. *BPM CBOK® – Business Process Management BPM Common Body of Knowledge, Version 3.0*. 2. Aufl. Gießen: Verlag Schmidt, 2015. ISBN: 978-3-921313-91-6.

[29] F. Puppe. *Einführung in Expertensysteme*. Berlin: Springer-Verlag, 1991. ISBN: 978-3-540-54023-6.

[30] G. Schreiber, H. Akkermans, A. Anjewierden et al. *Knowledge Engineering and Knowledge Management - The CommonKADS Methodology*. The MIT Press, 2000.

[31] J. Baumeister. *Agile Development of Diagnostic Knowledge Systems*. Berlin: Akademische Verlagsgesellschaft Aka GmbH, 2004. ISBN: 3-89838-284-2.

[32] J. Bloch. *Effective Java Programming Language Guide*. Amsterdam: Addison-Wesley, 2001. ISBN: 978-0201310054.

[33] J. Drawehn, M. Kochanowski, F. Kötter. *Business Process Management Tools 2014 - Marktüberblick*. Hrsg. von A. Weisbecker, J. Drawehn. Stuttgart: Fraunhofer Verlag, 2014. ISBN: 978-3-8396-0776-3.

[34] J. Plachetka. „Entwurf und Realisierung eines wissensbasierten Glukose-Management Systems mit d3web“. Bachelorarbeit. Hochschule Mannheim, Fakultät für Informatik, 2016.

[35] L.J. Bass, P.C. Clements, R. Kazman. *Software Architecture in Practice*. Addison-Wesley, 1998.

[36] M. Awad, J. Kuusela, J. Ziegler. *Object-Oriented Technology for Real-Time Systems: a practical approach using OMT and Fusion*. Prentice Hall, Juli 1996. ISBN: 0132279436.

[37] M. Dumas, M. La Rosa, J. Mendling, H. Reijers. *Fundamentals of Business Process Management*. 2. Aufl. Berlin: Springer-Verlag, 2018. ISBN: 978-3-662-56508-7.

[38] M. Fowler. *Patterns of Enterprise Application Architecture*. 1. Aufl. Pearson International, 2002. ISBN: 978-0321127426.

[39] M. Fowler. *UML konzentriert*. 3. Aufl. Addison-Wesley, 2004. ISBN: 978-3827321268.

[40] P.H. Winston. *Artificial Intelligence*. Massachusetts: Addison-Wesley, 1984. ISBN: 978-0-201-08259-3.

[41] S. Baier. „Entwicklung eines Augmentierungs-Editors für SmartCare Applikationen“. Bachelorarbeit. Universität zu Lübeck, Institut fur Medizinische Informatik, 2014.

[42] S. Behrens, Hrsg. *Das Einsparpotenzial innovativer Medizintechnik im Gesundheitswesen*. 1. Aufl. Spectaris e.V., Nov. 2006. ISBN: 3-930376-48-2.

[43] S. Liang. *The Java Native Interface: Programmer's Guide and Specification*. 1. Aufl. Addison Wesley, 1999. ISBN: 978-0201325775.

[44] S. Russell, P. Norvig. *Künstliche Intelligenz - Ein moderner Ansatz*. 2. Aufl. Prentice Hall, 2004. ISBN: 978-3-8273-7089-1.

[45] T. Debevoise, J. Taylor. *Prozess- und Entscheidungsmodellierung in BPMN/DMN*. CreateSpace, 2016. ISBN: 978-1519542960.

[46] U. Reimer. *Einführung in die Wissensrepräsentation: Netzartige und schema-basierte Repräsentationsformate*. Teubner, Stuttgart, 1991.

[47] V. Chandrasekhara, A. Eberhart, H.Hellbrück et al. *Java 5.0. Konzepte, Grundlagen und Erweiterungen in 5.0*. München: Carl Hanser Verlag, 2004. ISBN: 978-3446229464.

[48] W.J. Kox, C.D. Spies. *Check-up Anästhesiologie*. 2. Aufl. Heidelberg: Springer, 2005. ISBN: 978-3-7945-2995-7.

BUCHBEITRÄGE

[49] R.L. Riley. „Development of the three-compartment model for dealing with uneven distribution“. In: Hrsg. von J.B. West. Bd. 1. New York: Academic, 1980. Kap. Pulmonary Gas Exchange. Ventilation, BloodFlow, and Diffusion, S. 67–85.

[50] T. Rotter, R.B. de Jong, S.E. Lacko et al. „Improving healthcare quality in Europe: Characteristics, effectiveness and implementation of different strategies“. In: Hrsg. von R. Busse, N. Klazinga, D. Panteli et al. Health Policy Series 53. Copenhagen: European Observatory on Health Systems und Policies, 2019. Kap. Clinical pathways as a quality strategy.

[51] W. Oczenski. „Atemsysteme für die interoperative Beatmung“. In: Hrsg. von Atmen - Atemhilfen. Atemphysiologie und Beatmungstechnik. Thieme-Verlag, 2017. Kap. 15.7.

[52] W. Swoboda. „IT-unterstützte Modellierung klinischer Prozesse“. In: *Dienstleistungsmanagement im Krankenhaus II: Prozesse, Produktivität, Diversität*. Hrsg. von R.B. Bouncken, M.A. Pfannstiel, A.J. Reuschl. Wiesbaden: Springer Fachmedien Wiesbaden, 2014, S. 81–102. ISBN: 978-3-658-05134-1. DOI: 10.1007/978-3-658-05134-1_4.

ARTIKEL AUS FACHZEITSCHRIFTEN

[53] A. Battisti, J. Roeseler, D. Tassaux, P. Jolliet. „Automatic adjustment of pressure support by a computer-driven knowledge-based system during noninvasive ventilation: a feasibility study“. In: *Intensive Care Medicine* Bd. 32, Nr. 10 (Okt. 2006), S. 1523–1528. DOI: 10.1007/s00134-006-0267-6.

[54] A. Tripathi. „Low-Code/No-Code Development Platforms“. In: *International Journal of Computer Applications (IJCA)* Bd. 4, Nr. 1 (2023), S. 27–35.

[55] Arbeitsgemeinschaft der Wissenschaftlichen Medizinischen Fachgesellschaften (AWMF) - Ständige Kommission Leitlinien. „AWMF-Regelwerk „Leitlinien““. In: *Auflage 2.1* (2023). URL: https://www.awmf.org/regelwerk.

[56] B. Andersen, M. Kasparick, H. Ulrich, S. Franke, J. Schlamelcher, M. Rockstroh, J. Ingenerf. „Connecting the clinical IT infrastructure to a service-oriented architecture of medical devices“. In: *Biomedizinische Technik. Biomedical Engineering* Bd. 63, Nr. 1 (2018), S. 57–68. DOI: 10.1515/bmt-2017-0021.

[57] B. Hillmann, D. Schwarzkopf, T. Manser et al. „Structure and concept of ICU rounds: the VIS-ITS survey“. In: *Med Klin Intensivmed Notfmed* Bd. 117 (2022), S. 276–282. DOI: 10.1007/s00063-021-00830-3.

[58] B.G. Buchanan, E.A. Feigenbaum. „Dendral and meta-dendral: Their applications dimension“. In: *Artificial Intelligence* Bd. 11, Nr. 1-2 (Aug. 1978), S. 5–24.

[59] C. Fleischmann-Struzek, D. Schwarzkopf, K. Reinhart. „Inzidenz der Sepsis in Deutschland und weltweit: Aktueller Wissensstand und Limitationen der Erhebung in Abrechnungsdaten“. In: *Medizinische Klinik, Intensivmedizin und Notfallmedizin* Bd. 117, Nr. 4 (2022), S. 264–268. DOI: 10.1007/s00063-021-00777-5.

[60] C. Muche-Borowski, I. Kopp. „Wie eine Leitlinie entsteht“. In: *Z Herz-Thorax-Gefäßchir* Bd. 25 (2011), S. 217–223. DOI: 10.1007/s00398-011-0860-z.

[61] C.S. Scheer, E.J. Giamarellos-Bourboulis, R. Ferrer et al. „Status of Sepsis Care in European Hospitals: Results from an International Cross-Sectional Survey“. In: *American Journal of Respiratory and Critical Care Medicine* Bd. 521, Nr. 4 (Apr. 2025), S. 587–599. DOI: 10.1164/rccm.202406-1167OC.

[62] D. Berwick, T. Nolan, J. Whittington. „The Triple Aim: Care, Health, and Cost“. In: *Health affairs (Project Hope)* Bd. 27 (Mai 2008), S. 759–769. DOI: 10.1377/hlthaff.27.3.759.

[63] D. Schädler, G. Miestinger, T. Becher, I. Frerichs, N. Weiler, C. Hörmann. „Automated control of mechanical ventilation during general anaesthesia: study protocol of a bicentric observational study (AVAS)“. In: *BPMJ Open* Bd. 7, Nr. e014742 (2017). DOI: 10.1136/bmjopen-2016-014742.

[64] D.I. De Silva, W.A.C. Pabasara, S.V. Sangkavi et al. „The Effectiveness of Code Reviews on Improving Software Quality: An Empirical Study“. In: *International Journal of Recent Technology and Engineering* Bd. 12, Nr. 2 (Juli 2023). ISSN: 2277-3878.

[65] Das europäische Parlament und der Rat der europäischen Union. „Verordnung (EU) 2017/745 des europäischen Parlaments und des Rates über Medizinprodukte“. In: *Amtsblatt der Europäischen Union* Bd. L 117/1 (2017).

[66] E.H. Shortliffe. „Computer-based medical consultations: MYCIN“. In: *Journal of Clinical Engineering* Bd. 388, Nr. 1 (Okt. 1976). DOI: 10.1097/00004669-197610000-00011.

[67] E.W. Ely, A.M. Baker, D.P. Dunagan et al. „Effect on the duration of mechanical ventilation of identifying patients capable of breathing spontaneously“. In: *N Engl J Med* Bd. 335 (1996), S. 1864–1869.

[68] F.M. Brunkhorst, M.A. Weigand, M. Pletz et al. „S3-Leitlinie Sepsis – Prävention, Diagnose, Therapie und Nachsorge“. In: *Med Klin Intensivmed Notfmed (Suppl 2)* Bd. 115 (2020), S. 37–109. DOI: 10.1007/s00063-020-00685-0.

[69] Fachgesellschaften, Federführende and Fachgesellschaften, Beteiligte. „S3-Leitlinie Analgesie, Sedierung und Delirmanagement in der Intensivmedizin (DAS-Leitlinie 2020)“. In: *AWMF-Registernummer 001/012* (2020).

[70] G. Decker, J. Freund, A. Lübbe et al. „BPMN 2.0 - Business Process Model and Notation“. In: *Berliner BPM-Offensive* (2010). URL: https://bpm-conference.org/assets/docs/bpmn-poster/BPMN2_0_Poster_DE.pdf.

[71] G. Elke, W.H. Hartl, K.G. Kreymann et al. „DGEM-Leitlinie: Klinische Ernährung in der Intensivmedizin“. In: *Aktuelle Ernährungsmedizin* Bd. 43 (2018), S. 341–408.

[72] G. Marckmann, J. Schildmann. „Qualität und Ethik in der Gesundheitsversorgung“. In: *Bundesgesundheitsblatt* Bd. 65 (2022), S. 335–341. DOI: 10.1007/s00103-022-03492-4.

[73] G. Miestinger, J. Leitner, D. Schädler, P. Engl, C. Hörmann. „Automatische Steuerung der Beatmung während einer Vollnarkose (AVAS-Studie)“. In: *AINS - Anästhesiologie · Intensivmedizin · Notfallmedizin · Schmerztherapie* Bd. 54, Nr. S01 (2019), S2–S3. DOI: 10.1055/s-0039-1700681.

[74] G.P. Marelich, S. Murin, F. Battistella et al. „Protocol Weaning of Mechanical Ventilation in Medical and Surgical Patients by Respiratory Care Practitioners and Nurses, Effect on Weaning Time and Incidence of Ventilator-Associated Pneumonia“. In: *Chest* Bd. 118 (2000), S. 459–467.

[75] J. Allen. „Maintaining Knowledge about Temporal Intervals“. In: *CACM* Bd. 26, Nr. 11 (1983), S. 832–843.

[76] J. Angele, D. Fensel, D. Landes, R. Studer. „Developing Knowledge-Based Systems with MIKE“. In: *Automated Software Engineering* Bd. 5 (Jan. 1998), S. 389–418. DOI: 10.1023/A:1008653328901.

[77] J. Baumeister, J. Reutelshoefer, F. Puppe. „KnowWE: A Semantic Wiki for Knowledge Engineering“. In: *Journal of Applied Intelligence* Bd. 35 (2011), S. 323–344. DOI: 10.1007/s10489-010-0224-5.

[78] J. Kretschmer, A. Riedlinger, T. Becher, D. Schädler, N. Weiler, K. Möller. „A Family of Physiological Models to Simulate Human Gas Exchange“. en. In: *Biomedical Engineering / Biomedizinische Technik* Bd. 58.2013, Nr. S1 (2013). ISSN: 1862-278X. DOI: 10.1515/bmt-2013-4353.

[79] K. Eisentraut, D. Ammon, M. Winkler, V. Detschew. „Abbildung klinischer Behandlungspfade als Modelle in der Unified Modeling Language“. In: *Deutsche Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie* (2008).

[80] K.K. Giuliano, M. Lecardo, L. Staul. „Impact of protocol watch on compliance with the surviving sepsis campaign“. In: *American Journal of Critical Care* Bd. 20, Nr. 4 (2011), S. 313–321. DOI: 10.4037/ajcc2011421.

[81] L. Ohno-Machado, J.H. Gennari, S.N. Murphy et al. „The guideline interchange format: a model for representing guidelines“. In: *J Am Med Inform Assoc.* Bd. 5, Nr. 4 (1998), S. 357–372. DOI: 10.1136/jamia.1998.0050357.

[82] M. Botta, E.F.E. Wenstedt, A.M. Tsonas et al. „Effectiveness, safety and efficacy of INTELLiVENT–adaptive support ventilation, a closed–loop ventilation mode for use in ICU patients – a systematic review“. In: *Expert Review of Respiratory Medicine* Bd. 15, Nr. 1 (2021), S. 1403–1413. DOI: 10.1080/17476348.2021.1933450.

[83] M. Dojat, F. Pachet, Z. Guessoum et al. „NéoGanesh: a Working System for the Automated Controlof Assisted Ventilation in ICUs“. In: *Art Intell Med* Bd. 11 (1997), S. 97–117.

[84] M. Idrees, F. Aslam. „A Comprehensive Survey and Analysis of Diverse Visual Programming Languages“. In: *VFAST Transactions on Software Engineering* Bd. 10, Nr. 2 (2022), S. 47–60.

[85] M.E. Conway. „Proposal for an UNCOL“. In: *Communications of the ACM* Bd. 1, Nr. 10 (1958), S. 5–8. DOI: 10.1145/368924.368928.

[86] Medical Device Coordination Group. „MDCG 2020-1: Guidance on Clinical Evaluation (MDR) / Performance Evaluation (IVDR) of Medical Device Software“. In: (März 2020). URL: https://ec.europa.eu/docsroom/documents/40323.

[87] OECD. „Health at a Glance 2023: OECD Indicators“. In: *OECD Publishing* (2023). ISSN: 978-92-64-95793-0. DOI: 10.1787/7a7afb35-en.

[88] P. Badakhshan, K. Conboy, T. Grisold, J. vom Brocke. „Agile business process management: A systematic literature review and an integrated framework“. In: *Business Process Management Journal* Bd. 26 (Nov. 2019), S. 1505–1523. DOI: 10.1108/BPMJ-12-2018-0347.

[89] P. Jouvet, C. Farges, G. Hatzakis et al. „Weaning children from mechanical ventilation with a computer-driven system (closed-loop protocol): A pilot study“. In: *Pediatr Crit Care Med* Bd. 8, Nr. 5 (2007), S. 425–432. DOI: 10.1097/01.PCC.0000282157.77811.F9.

[90] P. Jouvet, P. Hernert, M. Wysocki. „Development and implementation of explicit computerized protocols for mechanical ventilation in children“. In: *Ann. Intensive Care* Bd. 1, Nr. 51 (2011). DOI: 10.1186/2110-5820-1-51.

[91] P.A. de Clercq, J.A. Blom, H.H.M. Korsten et al. „Approaches for creating computer-interpretable guidelines that facilitate decision support“. In: *Art Intell Med* Bd. 31 (2004), S. 1–27. DOI: 10.1016/j.artmed.2004.02.003.

[92] R. Luehmann. „Bericht über den Ersten Lübecker Workshop Evidence based Medicine“. In: *Deutsches Cochrane Centrum (ed.) Rundbrief Nr. 3 der Cochrane Collaboration Deutschland* (1997), S. 7–9.

[93] R. von Haken, M. Gruß, K. Plaschke, M. Scholz, R. Engelhardt, A. Brobeil1, E. Martin, M.A. Weigand. „Delir auf der Intensivstation“. In: *Der Anaesthesist* Bd. 59 (Feb. 2010), S. 235–247. DOI: 10.1007/s00101-009-1664-3.

[94] R.D. Branson, L. Berra, D.R. Hess. „Remembering the Explorers: Robert M. Kacmarek, PhD, RRT, FCCM“. In: *Critical Care Explorations* Bd. 3, Nr. 8 (2021). DOI: 10.1097/CCE.0000000000000510.

[95] R.L. Chatburn, S. Deem. „Should Weaning Protocols be used with all Patients who receive Mechanical Ventilation?“ In: *Respiratory Care* Bd. 52, Nr. 5 (2007), S. 609–621.

[96] S. Strahringer, M. Westner. „Low-Code/No-Code – Demokratisierung der IT?“ In: *HMD Praxis der Wirtschaftsinformatik* Bd. 61 (2024), S. 1067–1069. DOI: 10.1365/s40702-024-01108-w.

[97] S.M. Burns, S. Earven, C. Fisher et al. „Implementation of an institutional program to improve clinical and financial outcomes of mechanically ventilated patients: One-year outcomes and lessons learned“. In: *Crit Care Med* Bd. 31, Nr. 12 (2003), S. 459–467.

[98] T. Chaudhary, C. Hohenstein, O. Bayer. „Die goldene Stunde der Sepsis“. In: *Medizinische Klinik - Intensivmedizin und Notfallmedizin* Bd. 02 (2014).

[99] T. Pasch. „Die Zukunft der Anästhesiologie: vier Säulen oder mehr?“ In: *Anästh Intensivmed* Bd. 51 (2010), S. 550–558.

[100] T.A. Pryor, G. Hripcsak. „The Arden Syntax for Medical Logic Modules“. In: *International Journal of Clinical Monitoring and Computing* Bd. 10 (1993), S. 215–224.

[101] Y. Shahar, S. Miksch, P. Johnson. „The Asgaard project: a task-specific framework for the application and critiquing of time-oriented clinical guidelines“. In: *Artif Intell Med.* Bd. 14, Nr. 1-2 (1998), S. 29–51. DOI: 10.1016/s0933-3657(98)00015-3.

[102] ZaeFQ. „Das Leitlinien-Manual von AWMF und ÄZQ“. In: *Z. ärztl. Fortbild. Qual.sich. (ZaeFQ)* Bd. 95, Suppl. I (2001). URL: http://www.urbanfischer.de/journals/zaefq.

KONFERENZBEITRÄGE

[103] C. Flechsig, J. Lohmer, R. Lasch. „Realizing the Full Potential of Robotic Process Automation Through a Combination with BPM“. In: *Logistics Management*. Hrsg. von C. Bierwirth, T. Kirschstein, D. Sackmann. Springer International Publishing, 2019, S. 104–119. ISBN: 978-3-030-29821-0.

[104] M. Peleg, A.A. Boxwala, O. Ogunyemi et al. „GLIF3: the evolution of a guideline representation format“. In: *Proc AMIA Symp*. 2000, S. 645–649.

[105] R. Hatko, J. Reutelshoefer, J. Baumeister, F. Puppe. „DiaFlux: A Graphical Language for Computer-Interpretable Guidelines“. In: *Lecture Notes in Computer Science*. 2011. DOI: 10.1007/978-3-642-27697-2_7.

[106] R. Moen, C. Norman. „Evolution of the PDCA Cycle“. In: *7th ANQ Congress, Tokyo*. 7th ANQ Congress, Tokyo. Asian Network for Quality, Sep. 2009.

[107] R.Hatko, J. Baumeister, V. Belli, F. Puppe. „Modeling of Diagnostic Guideline Knowledge in Semantic Wikis“. In: *Proceedings of the Workshop on Open Knowledge Models (OKM-2010) at the 17th International Conference on Knowledge Engineering and Knowledge Management (EKAW)*. 2010.

NORMEN

[108] *DIN EN ISO 14155:2021-05 - Klinische Prüfung von Medizinprodukten an Menschen - Gute klinische Praxis*. DIN EN ISO, Mai 2021. DOI: 10.31030/3249461.

[109] *ISO/IEC 19510:2013 - Information Technology - Object Management Group: Business Process Model and Notation*. IEC, Juli 2013. DOI: 10.3403/30270926.

[110] *IEC62304:2006 - Medical device software - Software life cycle processes*. IEC, Mai 2006.

[111] *ISO IEC 25010:2023-11 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Product quality model*. ISO IEC, Nov. 2023.

[112] *DIN EN ISO 9000:2015-11 - Qualitätsmanagementsysteme - Grundlagen und Begriffe*. DIN EN ISO, Nov. 2015. DOI: 10.31030/2325650.

INTERNET-QUELLEN

[113] Ärztliche Zentralstelle für Qualitätssicherung in der Medizin. *Ärztliches Zentrum für Qualität in der Medizin (ÄZQ)*. Internet. [abgerufen am 16.06.2025]. URL: https://web.archive.org/web/20241202102030/https://www.aezq.de/.

[114] Cochrane-Deutschland. *Evidenzbasierte Medizin*. Internet. [abgerufen am 16.06.2025]. URL: www.cochrane.de/ueber-uns/evidenzbasierte-medizin.

[115] Fraunhofer-Institut für Experimentelles Software Engineering IESE. *Generative KI im Software Engineering: Szenarien und künftige Herausforderungen*. Internet. [abgerufen am 16.06.2025]. URL: www.iese.fraunhofer.de/blog/generative-ki-softwareentwicklung.

[116] Fraunhofer-Institut für Experimentelles Software Engineering IESE. *KI-gestütztes Software Engineering*. Internet. [abgerufen am 16.06.2025]. URL: www.iese.fraunhofer.de/de/trend/ki-software-engineering.html.

[117] Gabler Wirtschaftslexikon. *Definition Prozess*. Internet. [abgerufen am 16.06.2025]. URL: https://wirtschaftslexikon.gabler.de/definition/prozess-45614.

[118] Guidelines International Network. *International Guidelines Library*. Internet. [abgerufen am 16.06.2025]. URL: https://g-i-n.net/international-guidelines-library.

[119] Hamilton Medical AG. *INTELLiVent-ASV*. Internet. [abgerufen am 12.02.2026]. URL: https://www.hamilton-medical.com/de_DE/Products/Technologies/INTELLiVENT-ASV.html.

[120] IEEE Standards Association. *P11073-10107 - Standard for Nomenclature for External Control of Medical Devices*. Internet. [abgerufen am 16.06.2025]. URL: https://standards.ieee.org/ieee/11073-10107/7536/.

[121] Lehrstuhl für Informatik VI der Universität Würzburg. *d3web - The Open-Source Diagnostic Platform*. Internet. [abgerufen am 17.03.2026]. URL: https://www.d3web.de.

[122] Statistisches Bundesamt. *Gesundheitsausgaben im Jahr 2022 auf knapp 500 Milliarden Euro gestiegen*. Internet. [abgerufen am 16.06.2025]. URL: www.destatis.de/DE/Presse/Pressemitteilungen/2024/04/PD24_167_236.html.

[123] Wind River. *Wind River launches JWorks*. Internet. [abgerufen am 17.03.2026]. URL: https://www.windriver.com/news/press/news-399.

[124] World Health Organisation. *WHO Guidelines*. Internet. [abgerufen am 16.06.2025]. URL: www.who.int/publications/who-guidelines.