Entwurf und Transformationskonzepte für flexible klinische Workflow Modelle

Markus Bandt[1] Robert Kühn[2] Peter Forbrig[1]
Sebastian Schick[1] Ilvio Bruder[2] Andreas Heuer[1]

[1]Universität Rostock
{mb, schick, pforbrig, heuer}@informatik.uni-rostock.de

[2]IT Science Center Rügen
{kuehn, bruder}@it-science-center.de

Zusammenfassung

Prozesse der realen Welt werden in der Informationstechnologie oft in Form von Workflows abgebildet. Besondere Anforderungen stellen hierbei hochflexible klinische Prozesse (insbesondere im und um dem OP-Saal) aufgrund von Notfällen, Ausfällen, Komplikationen und Verschiebungen von Personal, Räumlichkeiten, Geräten und Material dar. Zur Beschreibung solcher Eigenschaften ist eine Workflow-Sprache mit Flexibilisierungskonzepten erforderlich. Darüber hinaus stellt sich der Entwurf solcher Workflows mit teilweise schwierig zu erfassenden Parametern sehr aufwendig und fehlerbehaftet dar. Der Entwurf mittels HOPS und YAWL sowie die Analyse und Umsetzung von adäquaten Flexibilisierungskonzepten, im Rahmen des Projektes PERIKLES, ist Kernthema dieses Artikels.

Schlüsselwörter

HOPS, YAWL, workflowbasiertes Assistenzsystem, Workflows

1. EINLEITUNG

Workflow-Management-Methoden sind eine bewährte Möglichkeit Vorgänge der realen Welt informationstechnisch zu erfassen. Besonders geeignet sind dabei natürlich exakt strukturierte Abläufe, wie bspw. Arbeits- und Geschäftsprozesse in der Industrie. Aber auch in ungenauen, hochdynamischen und flexiblen Umgebungen, wie dem hier gewählten Einsatzfeld perioperativer klinischer Prozesse, ist Workflow-Management geeignet.

Im Umfeld eines operativen Zentrums einer Klinik hat man es mit komplexen Koordinationsvorgängen zu tun. Dabei spielen das zeitliche sowie räumliche Management von Personal, Räumen, Geräten und Material eine wesentliche Rolle. Dynamische Änderungen der Arbeitsabläufe sind aufgrund von Notfällen, Ausfällen, unvorhergesehenen Komplikationen und Verschiebungen notwendig.

Um diese Herausforderungen adäquat umsetzen zu können, werden in diesem Artikel die Aspekte des Entwurfs eines solchen Assistenzsystems sowie geeignete Flexibilisierungskonzepte für das Workflow-Management vorgestellt und diskutiert. Im folgenden sollen diese Aspekte anhand des Projektes PERIKLES (Unterstützung **peri**operativer klinischer Prozesse durch **k**ooperierende f**l**exible Workflows und AutoID-**S**ensorsysteme) vorgestellt werden. Als Modellierungssprache dient HOPS (Higher-Order Process Specification), zur Spezifikation der Workflows wird YAWL (Yet Another Workflow Language) genutzt. Auf Basis von YAWL sollen benötigte Flexibilisierungskonzepte analysiert und umgesetzt werden.

Das Ziel von PERIKLES ist die Bereitstellung eines workflowbasierten Assistenzsystems, um das OP-Management zu entlasten und wichtige Aufgaben wie Planung, Koordinierung, Kommunikation und Überwachung des perioperativen Prozesses zu vereinfachen.

2. ENTWURF MIT HOPS

Vorgehensweise. Ein wesentlicher Teil von PERIKLES, ist die Aufgabenanalyse, welche sich an dem szenariobasierten Ansatz von Carroll & Rosson [3] orientiert. Ziel der Aufgabenanalyse ist es, die Arbeitsabläufe im perioperativen Prozess zu verstehen und die beteiligten Akteure bzw. deren Aufgaben zu beobachten. Dies ist wichtig, da der perioperative Prozess aus sehr vielen komplexen Arbeitsschritten besteht, bei denen viele Akteure miteinander interagieren. [6] beschreibt, dass sich Model-based Design (MDB) als Vorgehensweise und HOPS als Modellierungssprache gut für die Beschreibung von nebenläufigen, kooperierenden Prozessen eignet. Die Vorgehensweise und die Beschreibungsmittel der Anforderungsanalyse ist in Abbildung 1 dargestellt. Die linke Spalte beschreibt dabei die aktuelle, die rechte Seite die zukünftige Situation. Der erste Schritt beinhaltet die Beschreibung der Ist-Situation (Abbildung 1, links unten) und wird im Uhrzeigersinn weiter verfeinert. Für die Analyse werden vor allem Literatur, Interviews und Hospitationen eingesetzt. Die aktuelle Situation wird mit Hilfe von HOPS-Modellen, Szenarien, Hospitationsprotokollen, etc. beschrieben. Die in diesem Schritt erlangten Erkenntnisse werden danach verallgemeinert, indem man z. B. von den interviewten Akteuren bzw. der hospitierten Einrichtung abstrahiert

Das Projekt PERIKLES wird vom Bundesministerium für Bildung und Forschung (BMBF) unter dem Förderkennzeichen „01S09009“ gefördert.

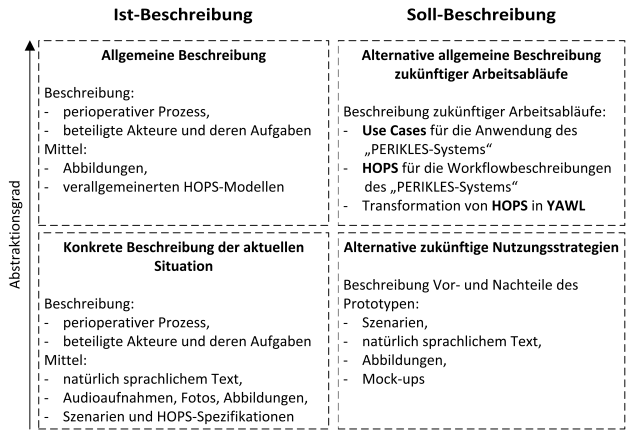

Abbildung 1: Vorgehensweise und Beschreibungsmittel für die Anforderungsanalyse (nach [4])

(Abbildung 1, links oben). Ziel ist eine sehr detaillierte Beschreibung des perioperativen Prozesses. Dafür ist ein intensiver Austausch mit den Akteuren aus dem perioperativen Prozess notwendig, da so Fehler frühestmöglich behoben werden können. Der nächste Schritt (Abbildung 1, rechts oben) ist die Erstellung von abstrakten Soll-Beschreibungen des perioperativen Prozesses. Hierfür wird das Ist-Modell in ein Soll-Modell („was sein könnte“) überführt. Anschließend wird das entstandene Modell von HOPS in YAWL transformiert (siehe Abschnitt 3). Zusätzlich werden die vorgesehenen Interaktionsmöglichkeiten der Akteure mit dem Assistenzsystem durch Use Cases beschrieben. Der letzte Schritt ist die Erstellung des Prototypen (Abbildung 1, rechts unten). Dabei wird das im vorherigen Schritt beschriebene Modell implementiert.
Eine alternative Vorgehensweise zu MDB ist der Model Driven Architecture (MDA) Ansatz. Anders als MDB verzichtet der MDA Ansatz auf die konkrete Beschreibung der Ist-Situation. Aus unserer Sicht ist jedoch dieser Schritt einer der wichtigsten. Dieser ermöglicht es, vorhandene Konflikte und Schwachstellen aufzudecken, die komplexen Arbeitsschritte zu verstehen und ein korrektes Modell der Ist-Situation zu entwerfen.

HOPS. HOPS ist eine universelle Spezifikationssprache zur Beschreibung der Interaktion zwischen kooperativen Systemen. Diese werden als Prozesse beschrieben. Ein Prozess P besteht aus mehreren Komponenten, bei denen es sich um eigenständige Prozesse handelt. Diese existieren parallel zu P und besitzen ein eigenständiges Verhalten. Darüber hinaus hat Prozess P mehrere atomare Operationen, mit denen das Verhalten des Systems beschrieben werden kann. Zusätzlich können Vor- oder Nachbedingungen spezifiziert werden, welche vor oder nach der Ausführung der Operation erfüllt sein müssen. Ein Prozess gliedert sich in mehrere Teilprozesse auf, welche aus einer Folge von Operationen oder Teilprozessen bestehen. Zusätzlich bietet HOPS mehrere temporale und strukturelle Operatoren an, welche die Ausführungsreihenfolge der Operationen beeinflussen können. So ist es möglich Teilverhalten eines Prozesses zu modellieren und modular zu einem Gesamtprozess zusammenzusetzen. Eine formale Einführung in das Konzept der Prozesse höherer Ordnung und eine ausführliche Beschreibung von HOPS wird in [5] gegeben.
HOPS ermöglicht es die formal als Prozess beschriebenen Systeme durch informale Beschreibungen wie z. B. Fotos, Abbildungen oder andere Texte anzureichern. Die angereicherten Modelle können animiert werden und bilden somit die Grundlage für die Diskussion mit den an der Analyse beteiligten Personen. Damit eignet sich HOPS sehr gut zur Beschreibung der Ist-Situation, da die Prozesse aus verschiedenen Perspektiven auf einen Sachverhalt modelliert werden können und deren Integration ausdrücken. Dank der textuellen Notation lassen sich leicht Änderungen in den Modellen vornehmen. In YAWL dagegen muss der Prozess in XML notiert bzw. mit dem verfügbaren Editor grafisch zusammengebaut werden. Das bedeutet einen enormen Mehraufwand und daher werden die Ist- und Soll-Modelle zuerst in HOPS entworfen und anschließend in YAWL transformiert.
Ein einfaches HOPS Beispiel wird in Abbildung 2 gezeigt. Dort wird der Prozess `Telefonieren` beschrieben. Das Verhalten des Prozesses ist in Zeile 11 festgelegt. Die Operationen sind in den Zeilen 4–8 definiert worden. `Telefonieren` besteht aus mehreren Teilprozessen `Waehlen`, `Sprechen` und `Auflegen` (Zeile 12–14). Der Teilprozess `Waehlen` (Zeile 12) besteht aus den Operationen `hoererAbnehmen`, `nummerWaehlen` und `warten`. Teilprozess `Sprechen` (Zeile 13) ist optional und wird nur ausgeführt, wenn die Gegenseite abnimmt.

```
1  PROCESS Telefonieren
2
3  OPS
4          hoererAbnehmen,
5          nummerWaehlen,
6          warten,
7          sprechen,
8          auflegen
9
10 SUB PROCESSES
11         Telefonieren = (Waehlen; [Sprechen]; Auflegen),
12         Waehlen = hoererAbnehmen; nummerWaehlen; warten,
13         Sprechen = sprechen,
14         Auflegen = auflegen
15
16 END PROCESS
```

Abbildung 2: HOPS-Prozess Telefonieren

3. MODELLTRANSFORMATION VON HOPS NACH YAWL

Wie in Abschnitt 2 beschrieben, eignet sich HOPS sehr gut für die Spezifikation der Ist- und Soll-Modelle. HOPS selbst ist jedoch kein Workflow-Management-System (WfMS). Es fehlen benötigte Funktionalitäten, wie z. B. eine Ressourcenverwaltung (für menschliche und nicht-menschliche Ressourcen). Um die in HOPS spezifizierten Prozesse für ein workflowbasiertes Assistenzsystem nutzen zu können, ist es notwendig diese in eine Workflow-Sprache zu überführen. Im Projekt PERIKLES fiel die Wahl dabei aus unterschiedlichen Gründen auf das WfMS YAWL (siehe Abschnitt 4). Ausschlaggebend war dabei unter anderem, dass YAWL grundsätzlich darauf ausgerichtet ist, Prozesse mit hoher menschlicher Beteiligung, wie sie z. B. im klinischen Bereich hauptsächlich auftreten, zu verwalten und zu steuern. Darüber hinaus ist es durch die Open Source Lizenz und die modulare, serviceorientierte Architektur des Systems möglich, gegebenenfalls eigene Erweiterungen zu entwickeln sowie bereits bestehende Systeme an das WfMS anzubinden.
Um die in HOPS beschriebenen Modelle in YAWL-Schemata

umzuwandeln, wird eine entsprechende Transformationssemantik benötigt. Die meisten HOPS-Konstrukte lassen sich dabei unmittelbar in YAWL überführen. Eine Sequenz in HOPS z. B. kann in eine YAWL Sequenz transformiert werden. Problematisch sind jedoch Konstrukte, die nicht direkt von YAWL unterstützt werden. Dazu gehört z. B. das Konstrukt „Alternative Aktivitäten“ (in HOPS `P = a [] b`, entweder Aktivität `a` oder Aktivität `b`. YAWL unterstützt dieses Konstrukt nicht direkt. Es gibt aber verschiedene Hilfskonstruktionen, die ein ähnliches Verhalten aufweisen, jedoch verschiedene Vor- oder Nachteile haben. Für das Problem der alternativen Aktivitäten wurden in YAWL verschiedene Lösungsmöglichkeiten herausgearbeitet. U. a. können Benutzereingaben (führe Aktivität `a` aus) oder *Cancelation Sets* (wenn `a` ausgeführt wurde, wird Aktivität `b` abgebrochen) verwendet werden. Bei den Benutzeraufgaben (realisierbar durch Verzweigungen oder den YAWL Worklet-Service) entscheiden die eingegebenen Daten und damit indirekt der Anwender, wie der weitere Arbeitsablauf ist. Im klinischen Bereich sind jedoch zusätzliche Nutzerinteraktionen mit dem System minimal zu halten bzw. häufig nicht erwünscht – die Arbeit am Patienten hat unbedingten Vorrang. Dieses kleine Beispiel zeigt, dass für einige Teilschemata keine eindeutige Transformation von HOPS in YAWL möglich ist. Um eine äquivalente Transformation zu gewährleisten, ist es notwendig einen entsprechenden semantischen „Regelkatalog“ aufzustellen. Mit Hilfe dieser Regeln wird jedem HOPS-Konstrukt ein passendes YAWL-Konstrukt zugeordnet. Bei eventuellen Änderungen im HOPS-Modell sind diese Änderungen eindeutig auf das YAWL-Schema übertragbar.

4. UMSETZUNG PERIOPERATIVER PROZESSE MIT YAWL

YAWL - Formale Grundlagen. Um dem Umstand Rechnung zu tragen, dass perioperative Prozesse hoch dynamisch und zum Teil nicht planbar sind, muss das verwendete Workflow-Management-System entsprechende Techniken bereitstellen. Hier stellt die flexible Anpassung der modellierten Arbeitsabläufe eine besondere Herausforderung dar. YAWL bietet eine Grundlage, mit der einige Flexibilisierungskonzepte bereits unterstützt werden. Für die Modellierung stellt die Sprache 14 Grundelemente bereit. Die *Atomic Task* bildet einzelne Arbeitsschritte für Menschen oder Maschinen ab. Die *Composite Task* bindet andere Prozesse ein. Mit Hilfe der *Condition* wird der Status eines Prozesses abgebildet. Jedem Arbeitsablauf sind außerdem eine *Input-* und *Output Condition* zugeordnet die jeweils den Beginn und das Ende eines Prozesses beschreiben. Eine *Multiple Instance Task* ermöglicht das gleichzeitige Ausführen mehrerer Instanzen einer Task. Neben den bereits genannten Elementen werden verschiedene Split- und Join- Typen unterstützt. Der *AND-Split* aktiviert alle ausgehenden Kanten und entsprechend wird der *AND-Join* über alle eingehenden Kanten aktiviert. Ähnlich verhält sich der *OR-Split*, wobei hier mindestens eine Kante aktiviert wird. Beim *OR-Join* muss mindestens eine der eingehenden Kanten aktiviert sein. Der *XOR-Split* aktiviert genau eine Kante und der *XOR-Join* wird durch genau eine Kante aktiviert.
Als weiteres Sprachelement bietet die *Cancelation Region* die Möglichkeit definierte Bereiche innerhalb eines Prozesses in Abhängigkeit einer Bedingung abzubrechen. Der Bereich kann sich auf eine atomare Task aber auch auf vollständige Netze beziehen.
YAWL bietet neben der Kontrollflussperspektive auch die Möglichkeit Daten zu modellieren und z.B. für den Kontrollfluss zu nutzen. Hierfür stehen eine Menge von Standard XML-Datentypen zur Verfügung. Wobei das Typsystem von YAWL XML-Schema verwendet, welches um eigene Typen erweitert werden kann. Variablen werden entweder einem Netz oder einer Task zugeordnet. Diese Zuordnung beschreibt zugleich auch den Sichtbarkeitsbereich der Daten. Task-Variablen sind immer als Abbildung von bzw. auf Netz-Variablen unter Verwendung der XML-Anfragesprachen XPath/XQuery beschrieben. Der interne Datentransfer findet immer vom Netz zur Task oder umgekehrt statt. Ein Transfer der Daten von einer Task zu anderen ist nicht erlaubt, da der Sichtbarkeitsbereich einer Task-Variablen lokal zur Task ist. Der externe Datentransfer muss immer über Task-Variablen durchgeführt werden. Neben dem Datenaustausch können die Daten auch für den Kontrollfluss verwendet werden. Hierbei besteht die Möglichkeit ausgehende Kanten eins *XOR-Splits* oder *OR-Splits* mit Bedingung auf Netz-Variablen zu erweitern. Hierbei werden dann entsprechend der Bedingungen nur einzelne Kanten aktiviert.

Flexibilität. YAWL unterstützt verschiedene Konzepte der Flexibilität. Während der Modellierungsphase kann durch Unterspezifikation (hier Late Binding) ein gewisser Grad an Flexibilität erreicht werden. Zur Ausführungszeit kann durch Exception-Handling und (eingeschränkt) Jumps Flexibilität auf Ebene der Instanzanpassungen erreicht werden.
Late Binding [13] wird in YAWL durch das Konzept der Worklets [1] umgesetzt. Dabei können über die, in YAWL vorhandene, Web-Service-Schnittstelle zur Laufzeit Worklets eingebunden werden. Ein Worklet beschreibt im einfachsten Fall eine atomare Workflow-Task oder eine vollständige Prozessbeschreibung. Anhand von Regeln kann ein Worklet ausgewählt werden. Die Ausführung wird einer speziellen Task im Workflow zugeordnet.
Das Exception-Handling [13] bietet die Möglichkeit auf unerwartete Ereignisse zur Laufzeit zu reagieren und wird durch einen vergleichbaren Ansatz wie bei dem vorher vorgestellten Late Binding umgesetzt. Die Exlets [2] bieten neben dem Einbinden von Worklets auch die Möglichkeit verschiedene Ausnahme-Typen zu definieren. Es werden Ereignisse erzeugt, wenn ein Case oder eine Task aufgerufen oder beendet wird. Anhand eines Regelsystems wird dann entschieden ob eine Ausnahme eingetreten ist. Um auf die verschiedenen Ausnahmen geeignet reagieren zu können, werden entsprechende Operationen angeboten. Dabei können einzelne Work-Items oder Cases gelöscht oder angehalten werden. Außerdem kann die Bearbeitung von Work-Items und Cases wieder fortgesetzt oder neu gestartet werden. Über die Operation *Compensate* können zum Beispiel Worklets eingebunden werden, um bereits ausgeführte Arbeitsschritte zu kompensieren.
Das Konzept der Jumps [9] bietet die Möglichkeit zur Laufzeit in den Kontrollfluss der Instanz einzugreifen und so auf Ausnahmesituationen zu reagieren. YAWL unterstützt dieses Konzept nur zum Teil. Es werden statische Forward-Jumps ermöglicht, wobei kein „Nachholen“ möglich ist. Eingeschränkte dynamische Forward-Jumps sowie eingeschränkte Backward-Jumps werden auch ermöglicht.

Lösungsansatz. Ein Schwerpunkt der Analysephase bestand darin, die spezifischen Anforderungen an Prozess-Flexibilität im klinischen Bereich zu erfassen [8]. Die Arbeit mit Patienten ist naturgemäß durch hohe Flexibilität gekennzeichnet, da jeder Patient einen individuellen Krankheitsverlauf aufweist und entsprechend behandelt werden muss. Schemata, die solche Behandlungsabläufe beschreiben, müssen daher abstrakt und gleichzeitig flexibel sein, um Allgemeingültigkeit für die jeweiligen therapeutischen Maßnahmen zu erreichen.
Im Verlauf der Transformation konnten die folgenden Klassen von Flexibilitätskonstrukten identifiziert werden, für die YAWL keine eindeutige Entsprechung bereitstellt.
Die Klasse der partiellen Ordnung von Aktivitäten kennzeichnet Prozesse, in denen einige Aktivitäten in festgelegter Reihenfolge ablaufen müssen, während andere (nichtoptionale) Aktivitäten an beliebiger Stelle in dieser Sequenz auftreten können. In YAWL können solche Workflows z. B. unter Verwendung von parallel geschalteten *Cancelation Regions* ausmodelliert werden. Allerdings gewinnt ein entsprechendes Workflow-Schema mit zunehmender Anzahl von Aktivitäten (sowohl von festen als auch variablen) bei diesem Ansatz sehr schnell an Komplexität. Eine mögliche Alternative, welche die Komplexität des Schemas nicht erhöht, bietet in diesem Zusammenhang der kooperierende Einsatz von deklarativen Systemen wie z. B. DECLARE [12]. YAWL bietet die Möglichkeit, solche Systeme als externe Dienste anzubinden.
Die Optionalität von Aktivitäten mit Spezifizierung zur Laufzeit ist eine weitere Problemstellung, für die es in YAWL kein unmittelbares Konstrukt gibt. Allerdings lässt sich mit den vorhandenen sprachlichen Mitteln das Konzept der Jumps dahingehend realisieren, dass Aktivitäten durch Auswertung von workflowinternen Daten oder Nutzereingaben übersprungen werden können.
Alternativen von Aktivitäten treten im perioperativen Kontext häufig auf und können in YAWL mit Hilfe einer Kombination von *XOR-Splits* und *Cancelation Regions* umgesetzt werden.
Die temporale Synchronisation von Aktivitäten wird insbesondere dann benötigt, wenn mehrere Aktivitäten unterschiedlicher Benutzer durchgeführt worden sein müssen bevor eine andere Aktivität beginnen kann (z. B. muss sowohl der Patient als auch der OP-Saal auf die Operation vorbereitet worden sein, bevor diese OP beginnen kann). Dieses Konstrukt lässt sich jedoch durch die Verwendung des Sprachelements *AND-Join* relativ einfach realisieren.
Der Ausfall von Aktivitäten, welcher zur Modellierungszeit noch nicht spezifiziert werden kann, ist eine weitere Problemklasse, für die es in YAWL derzeit keine direkte Sprachunterstützung gibt. Mögliche Lösungen stellen hier die Verwendung von Nutzereingaben oder gezieltes Exception-Handling dar.
Der Abbruch von Aktivitäten ist ein generelles Problem im Workflow-Management. Eine mögliche Lösung für den Einsatz von YAWL bietet hier ebenfalls das Exception-Handling, welches zusätzlich auch Möglichkeiten zur Kompensation bietet.
Erwartete und unerwartete Ausnahmen im weitesten Sinne können in YAWL generell durch Exception-Handling behandelt und umfassend kompensiert werden. Dies kann auch die Verwendung externer Dienste mit einschließen.

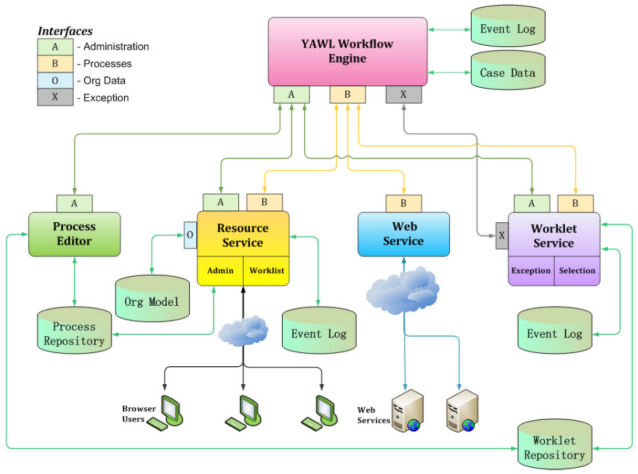

Abbildung 3: YAWL Architektur (abgewandelt aus [11])

5. SYSTEM

Anforderungen. Die bisher im Dokument behandelten Aspekte, vom Entwurf mit HOPS über die Transformation nach YAWL und die Flexibilitätsanforderungen im klinischen Bereich bis hin zu entsprechenden Lösungsansätzen in YAWL, sind grundlegende Bestandteile des bereits erwähnten Projekts Perikles. Grundsätzlich wird dabei der Ansatz eines workflowbasierten Assistenzsystems (im konkreten Fall für die Planung und Koordination von medizinischen Operationen) verfolgt, welches neben den zugrunde liegenden Modellen auch die Verwaltung limitierter Ressourcen, eine integrierte Ereignissverarbeitung sowie den Zugriff auf externe Datenquellen in die Prozesssteuerung einbindet. Insofern unterscheidet sich der Grundgedanke der Prozesssteuerung und -beobachtung von der aktuell üblichen Verfahrensweise im klinischen Bereich, die hauptsächlich darin besteht, den Behandlungsprozess über die medizinische Dokumentation zu steuern bzw. nachzuvollziehen.
Im konkreten Anwendungsszenario im perioperativen Bereich besteht neben den hohen Anforderungen an die Flexibilität der Abläufe auch der dringende Bedarf, kontextabhängig große Datenmengen aus unterschiedlichen Datenquellen (z. B. aus Datenbanken oder Diensten als Schnittstellen zu externen Systemen) bereit zu stellen. Zum Beispiel erreichen Patientenakten einen großen Informations- und Datenumfang und setzen sich unter Umständen aus Befunden mehrerer medizinischer Fachabteilungen (häufig sogar aus verschiedenen Einrichtungen) zusammen. Die Integration dieser Daten in den Prozess beginnt bei der Modellierung der Prozesse. Geeignete Beschreibungen der Daten und die Verknüpfung mit den Workflow-Beschreibungen sollen hier eine bessere Verfügbarkeit der Daten gewährleisten. Ein weiteres Problem ist die Heterogenität der Datenquellen. Wir schlagen ein allgemeines Konzept für die Integration unterschiedlicher Datenquellen im Bereich perioperativer Prozesse vor [10]. Dies erfordert unter anderem eine funktionelle Erweiterung der YAWL-Workflow-Engine, welche im folgende kurz vorgestellt werden soll.
YAWL Architektur. Die Architektur des WfMS YAWL entspricht weitgehend der in [7] durch die WfMC (Workflow Management Coalition) vorgestellte Architektur. Als wesent-

liche Änderung zur Architektur der WfMC zeigt sich die konsequent eingeführte service-orientierte Architektur (Abbildung 3). Dabei werden alle Komponenten, z. B. Ressourcen aber auch Arbeitslisten für Workflow-Teilnehmer, nur über entsprechende Service-Schnittstellen (Custom Services) zur Verfügung gestellt.

Zugriff auf externe Datenquellen und Dienste. Entsprechend der von der WfMC vorgestellten Architektur werden alle für einen Arbeitsablauf benötigten Daten innerhalb des Workflow-Repositories abgespeichert. Wie in Abschnitt 4 bereits beschrieben, stehen diese Daten dann als globale Case-Variablen über ein Mapping auch den einzelnen Tasks bereit. Im vorgestellten Anwendungskontext bringt diese Vorgehensweise aber verschiedene Probleme mit sich. Sollen die Daten innerhalb eines Arbeitsablaufes bereitgestellt werden, muss bereits in der Modellierungsphase das Modell derart erweitert werden, dass einzelne Tasks hinzugefügt werden, welche nur für den Datenaustausch zwischen Workflow-Repository und externen Datenquellen verwendet werden. Komplexe, unübersichtliche Modelle sind die Folge. Ein weiteres Problem wird aufgeworfen, wenn die externen Daten geändert werden und dies nicht an den Arbeitsablauf propagiert wird. Datenabhängigkeiten des Kontrollflusses haben zum Beispiel inkonsistente Workflow-Zustände zur Folge. Änderungen innerhalb einzelner Workflow-Tasks müssen außerdem in bestimmten Situationen sofort an externe Datenquellen weitergereicht werden.
Als Lösung schlagen wir vor, Workflow-Variablen an externe Datenquellen zu binden. Datenstrukturen werden mit Hilfe von XML dargestellt. Auf diesen Datenstrukturen sind dann Bedingungen durch XPath/XQuery Ausdrücke beschrieben. Der Inhalt der externen Datenquellen wird weiterhin als Kopie innerhalb des Workflow-Repositories verwaltet. Neben Integritätsbedingungen auf diesen Daten müssen zusätzlich Strategien für die Synchronisation eingeführt werden. Der Zeitpunkt, wann eine Kopie aktualisiert werden muss oder wann sie an die externe Datenquelle zurückgeschrieben wird kann hier angegeben werden. Weiterhin müssen geeignete Reaktionen auf verletze Integritätsbedingungen ermöglicht werden. Hier kommen zum Beispiel kompensierende Abläufe, Abbrüche ohne Kompensation, oder auch Alternativen zum Einsatz. Die im Umfeld klinischer Prozesse eingesetzten Datenquellen erstrecken sich über einfache tupelbasierte relationale Datenquellen bis hin zu nachrichtenbasierten Datenquellen wie HL7 Datenströmen.
Die YAWL-Workflow-Engine soll durch einen Custom-Service mit den vorgestellten Anforderungen erweitert werden. Die bereitgestellten Inferfaces B und X stellen hier eine geeignete Schnittstelle dar. Die jeweiligen Datenquellen werden dann über eine entsprechende Plug-In-Architektur integriert.

6. AUSBLICK

Die im Artikel beschriebenen Arbeiten haben gezeigt, dass kritische, klinische Prozesse mit ihrer hohen Flexibilität mittels Workflows umfassend beschreibbar sind. Dabei konnte hier ein adäquater Entwurfsprozess definiert und nötige Transformationen beschrieben werden. Die Umsetzung der Flexibilisierungskonzepte hat gezeigt, dass Workflow-Sprachen prinzipiell geeignet sind, aber dahingehend erweitert werden müssen.
Zukünftig muss der Entwurfsprozess konsequent bis zur laufenden Anwendung weiter entwickelt werden. Dazu steht im nächsten Schritt die Umsetzung der Workflows in einer Simulationsumgebung an. Auch hier ist wieder zu überprüfen, ob die Flexibilisierungskonzepte die realen Bedingungen abbilden können. Mit dem entwickelten Assistenzsystem sind anschließend auch Tests im klinischen Umfeld geplant.

7. LITERATUR

[1] M. Adams, A. H. M. ter Hofstede, D. Edmond, and W. M. P. van der Aalst. Worklets: A Service-Oriented Implementation of Dynamic Flexibility in Workflows. In R. Meersman and Z. Tari, editors, *OTM Conferences (1)*, volume 4275 of *Lecture Notes in Computer Science*, pages 291–308. Springer, 2006.

[2] M. Adams, A. H. M. ter Hofstede, W. M. P. van der Aalst, and D. Edmond. Dynamic, Extensible and Context-Aware Exception Handling for Workflows. In R. Meersman and Z. Tari, editors, *OTM Conferences (1)*, volume 4803 of *Lecture Notes in Computer Science*, pages 95–112. Springer, 2007.

[3] J. M. Carroll, M. B. Rosson, G. Chin, and J. Koenemann. Requirements Development in Scenario-Based Design. *IEEE Transactions on Software Engineering*, 24(12):1156–1170, 1998.

[4] A. Dittmar. Perikles Projektbericht zum Meilenstein 1, 2009.

[5] A. Dittmar and P. Forbrig. A unified description formalism for complex HCI-systems. *International Conference on Software Engineering and Formal Methods*, pages 342–351, 2005.

[6] A. Dittmar and P. Forbrig. Task-based design revisited. In *EICS '09: Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, pages 111–116, New York, NY, USA, 2009. ACM.

[7] D. Hollingsworth. The Workflow Reference Model - TC00-1003, 1995.

[8] T. Möller. Untersuchung zur Flexibilisierung klinischer Workflows. Diplomarbeit, Universtity of Rostock, Universtity of Rostock, 2009.

[9] M. Reichert, P. Dadam, and T. Bauer. Dealing with Forward and Backward Jumps in Workflow Management Systems. *Software and System Modeling*, 2(1):37–58, 2003.

[10] L. Song. Konzepte für die Integration von XML-Schema und Prozessbeschreibungen. Diplomarbeit, Universtity of Rostock, Universtity of Rostock, 2009.

[11] W. M. P. van der Aalst, M. Adams, A. H. M. ter Hofstede, M. Pesic, and H. Schonenberg. Flexibility as a service. In L. C. 0002, C. Liu, Q. Liu, and K. Deng, editors, *DASFAA Workshops*, volume 5667 of *Lecture Notes in Computer Science*, pages 319–333. Springer, 2009.

[12] W. M. P. van der Aalst, M. Pesic, and H. Schonenberg. Declarative workflows: Balancing between flexibility and support. *Computer Science - R&D*, 23(2):99–113, 2009.

[13] B. Weber, S. W. Sadiq, and M. Reichert. Beyond rigidity - dynamic process lifecycle support. *Computer Science - R&D*, 23(2):47–65, 2009.