Vorhaben	**SHIP** Semantic Integration of Heterogenous Processes
Titel	**Abschlussbericht**
Förderkennzeichen	01 IW 10002
Zuwendungsempfänger	Deutsches Forschungszentrum für Künstliche Intelligenz GmbH Trippstadter Straße 122, D-67663 Kaiserslautern
Ausführende Stelle	DFKI GmbH - Cyber-Physical Systems, Bremen
Projektleiter	Prof. Dr. Rolf Drechsler
Bewilligungszeitraum	1.1.2011 - 31.12.2013
Autoren	Dr. Serge Autexier, Prof. Dr. Dieter Hutter
Erstellungsdatum:	Mai 2014

Gefördert durch:

aufgrund eines Beschlusses
des Deutschen Bundestages

Das diesem Bericht zugrunde liegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 01 IW 10002 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt beim Autor.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API	Application Programming Interface
BAALL	Bremen Ambient Assisted Living Laboratory
CASL	Common Algebraic Specification Language
CoLOSS	Coalgebraic Logic Satisfiability Solver
DFKI	Deutsches Forschungszentrum für Künstliche Intelligenz
DIN	Deutsches Institut für Normung
DL	Dynamische Logik
HasCASL	Higher-Order Specification - Common Algebraic Specification Language
Hets	Heterogeneous Tool Set
KI	Künstliche Intelligenz
MPG	Medizinproduktegesetz
NP	Nichtdeterministisch polynomielle Zeit
PSPACE	Polynomieller Speicher
Rolland	Bremer Autonome Rollstuhl
SHIP	Semantic Heterogeneous Integration of Processes
SIMPLE	Semantisch fundierte Implementierung klinischer Leitlinien
TL	Temporallogik

Einleitung

Das vorliegende Dokument stellt den Abschlussbericht des Vorhabens SHIP dar. SHIP wurde durch das Bundesministerium für Bildung und Forschung (BMBF, Förderkennzeichen 01 IW 10002) gefördert.

Dieser Abschlussbericht folgt dabei dem in der Anlage 2 zum NKBF 98 gegebenen Muster. In Kapitel 1 wird eine kurze Darstellung des Projektes gemäß Punkt I gegeben, Kapitel 2 stellt die inhaltlichen Entwicklungen in einer eingehenden Darstellung gemäß Punkt II des Musters dar. Die Punkte III (Erfolgskontrollbericht) und IV (Berichtsblatt bzw. Document Control Sheet) werden durch gesondert abgegebene Dokumente erfüllt.

Die dargestellten Arbeiten umfassen alle vom DFKI geleisteten Arbeiten im Vorhaben SHIP inklusive dessen Aufstockungsprojektes SHIP4AAL. Abbildung 1.2 auf Seite 15 gibt den Überblick über die Aufteilung der Arbeitspakete.

1 Kurzdarstellung (NKBF 98 8.2 I)

In diesem Kapitel wird zunächst die Aufgabenstellung des Projektes beschrieben (Abschnitt 1.1). Anschließend werden die Voraussetzungen des Vorhabens (Abschnitt 1.2) sowie Planung und Ablauf des Vorhabens (Abschnitt 1.3) behandelt. Es folgen Erläuterungen zum wissenschaftlichen Stand (Abschnitt 1.4) und zur Zusammenarbeit mit anderen Stellen (Abschnitt 1.5).

1.1 Aufgabenstellung

Die mit dem Paradigma der serviceorientierten Architekturen begonnene Integration verschiedener Systeme hinsichtlich einer gemeinsamen Datensicht wird sich zunehmend auch auf die Komposition und Integration verschiedenartiger Arbeitsabläufe und Prozesse ausdehnen. Die wechselseitigen Abhängigkeiten der ablaufenden Prozesse und die Komplexität der durch sie verarbeiteten Informationen erschweren zunehmend die bisher übliche Vermittlung zwischen verschiedenartigen Prozessen. Das Projekt SHIP zielt auf eine semantisch basierte Koppelung, Integration und Verifikation heterogener Prozesse, wobei Abgleich und Integration der durch diese Prozesse verarbeiteten Informationen im Mittelpunkt steht. Der Bereich der Alters-gerechten Assistenzsysteme (AAL) dient als Anwendungsszenario für die in SHIP zu entwickelnden Techniken und Werkzeuge.

1.1.1 Technische Arbeitsziele

Das Projekt gliederte sich in drei Teilziele, die durch entsprechende Meilensteine abgedeckt wurden.

Meilenstein M1 (12/2011): Der erste Meilenstein umfasste zwei wesentliche Resultate. Bezüglich des Arbeitspakets AAL sollte die Anforderungsanalyse des BAALL-Szenarios abgeschlossen und die daraus folgenden Entwicklungsschritte in SHIP festgelegt sein. Aus technologischer Sicht sollten geeignete Modallogiken für die Spezifikation insbesondere der Anwendungsprozesse entwickelt und entsprechende Verifikationsverfahren ausgewählt werden. Weiterhin sollte eine allgemeine und einheitliche Sprache zur Verfeinerung heterogener Spezifikationen zur Verfügung gestellt werden, die es erlaubt, Verfeinerungen der Anwendungsdomäne in getrennte Verfeinerungen für Daten und Prozesse aufzuteilen.

Meilenstein M2 (12/2012): Zu diesem Meilenstein sollten die folgenden Resultate erreicht werden. Es sollten die abstrakten Prozesse und Daten spezifiziert und einzelne Eigenschaften auch bereits mit Hilfe der Beweisunterstützung aus der ersten Projektphase verifiziert sein. Eine elementare Plattform zur Kommunikation mit dem BAALL solte verfügbar sein, um Prozesse auszuführen und zu überwachen und um Sensoren und Applikationen anzusprechen. Aus technologischer Sicht sollte die Entwicklung der domänenspezifischen Logiken abgeschlossen sein, und eine entsprechende Werkzeugunterstützung für die Entwicklung und Analyse von Spezifikationen in diesen Logiken zur Verfügung stehen. Weiterhin sollte ein genereller Mechanismus für eine enge Integration von Prozessalgebren und Datenlogiken verfügbar sein. Die einheitliche Verfeinerungssprache sollte um die Behandlung von Sichten erweitert und damit als Grundlage für die Spezifikation von Daten- und Prozesskonnektoren dienen.

Meilenstein M3 (12/2013): Zum letzten Meilenstein M3 sollten alle übrigen Aufgaben abgeschlossen sein. Im Bereich Ambient Assisted Living sollte es eine Reihe verifizierter Prozesse auf verschiedenen Abstraktionsebenen geben, wie sie im BAALL zum Einsatz kommen. Weiterhin sollten im System Änderungen von Prozessen bzw. der BAALL-Umgebung unterstützt werden. Interaktive, shell-artige Umgebungen sollten eine Prozesssimulation durch symbolische Ausführung ermöglichen: Situationen sollten deklarativ spezifiziert, und Prozesse auf diesen Spezifikationen mit Hilfe der im vorangegangen Meilenstein entwickelten Abstraktionen operieren. Die zu entwickelnden Techniken zu Laufzeitaspekten von Abstraktionen, Verfeinerungen und Sichten sollten dazu eingesetzt werden, die verschiedenen Sichten und Abstraktionsebenen der zu verarbeitenden Daten in einem konsistenten Zustand zu halten.

1.1.2 Wissenschaftliche Arbeitsziele

In SHIP wurden semantische Modellierungs-, Analyse- und Verifikationstechniken erweitert, angepasst und integriert, um zuverlässige verteilte und heterogene Systeme zu entwerfen, zu implementieren, zu simulieren oder zu adaptieren. Ziel war die formale Modellierung heterogenen Systemprozesse, um die resultierenden Teillösungen intelligent in eine Gesamtlösung integrieren zu können, und dadurch die Kommunikation dieser Systeme mit ihrer Umwelt in ihren unterschiedlichen Sichten und Abstraktionsebenen semantisch zu repräsentieren und gegenseitig abzugleichen. Dabei ging es weniger darum, das interne Verhalten der Prozesse im Detail zu erfassen als vielmehr ihre Kommunikation mit der Umwelt in ihren unterschiedlichen Sichten und Abstraktionsebenen semantisch zu repräsentieren. Die erforderliche formale Modellierung zerlegt damit das System in verschiedene, orthogonale Sichten.

Die in diesem Projekt vorgesehene Methodik zur Behandlung dieser Probleme war die Bündelung und Verknüpfung verschiedener formaler Methoden für die Spezifikation, Verifikation, Simulation und Realisierung verteilter, kommunizierender Prozesse mit heterogenen Datenrepräsentationen. Dies umfasste sowohl Methoden zur Spezifikation der Konsistenz von Daten, die in unterschiedlichen Sichten und Darstellungen in Unterprozessen repräsentiert und verarbeitet werden als auch Methoden zur Abstraktion und Verfeinerung von Daten und Prozessen. Eine heterogene formale Spezifikation der Prozesse sollte als semantische Fundierung dienen, auf der anwendungsspezifische Analyse- und Optimierungswerkzeuge operieren, und als Basis, um formal die Einzelprozesse mit den Anforderungen aus dem jeweiligen Anwendungskontext zu verknüpfen. Die zentrale Technologie des Projektes bildete eine intelligente Integrierte Entwicklungsumgebung (IDE) für heterogene Daten und Prozesse. Vorhandene Dienste und Aktivitäten sollten zusammmen mit der abstrakten Spezifikation ihrer Ein- und Ausgabedaten zu elementaren Prozessblöcken integriert werden. Einzelne Prozessblöcke sollten mit Hilfe vorgefertigter Kompositionsprimitiven zu neuen komplexeren Prozessblöcken zusammengefügt werden.

Im Rahmen einer Aufstockungsphase SHIP4AAL sollten die Ergebnisse von SHIP für eine Integration eines Roboters in das BAALL verwendet werden, um die Praktikabilität des Ansatzes zu überprüfen. Insbesondere sollten die folgende Ziele erreicht werden: 1. Erweiterung des SHIP-Tool um eine Schnittstelle zu dem OpenURC-Standard, um eine technische Basisinfrastruktur für die semantische Integration von Diensten im BAALL und anderen AAL-Szenarien zu schaffen. 2. Prototypische Integration des im Rahmen des CAPIO-Projektes am Robotics Innovation Centre entwickelten Exoskeletts und des anthropomorphe Roboters AILA in das BAALL

unter Nutzung der SHIP-Methodologie. Hierbei soll AILA im Rahmen von SHIP als Demonstrator für den Einsatz einer Haushaltshilfe in Kombination mit einem Exoskelett verwendet werden. Ziel hierbei war es zu zeigen, wie ein tragbares Exoskelett mit Kraftrückkopplung (engl. Force-Feedback) dazu verwendet werden kann, einer nicht mehr vollständig mobilen Person die Möglichkeit zu eröffnen mit Hilfe eines Roboters Aufgaben im Haushalt selbstständig durchführen zu können. 3. Validation durch ein Demo-Szenario. Als Demonstrationsszenario war zum Zeitpunkt der Antragstellung angedacht, dass eine einfache Aufgabe in der Küche des BAALs durch eine im Wohnbereich sitzende Person ferngesteuert durchgeführt werden kann.

1.2 Voraussetzungen, unter denen das Vorhaben durchgeführt wurde

Das Deutsche Forschungszentrum für Künstliche Intelligenz GmbH mit Sitzen in Kaiserslautern, Saarbrücken und Bremen gehört auf dem Gebiet innovativer Softwaretechnologien zu den führenden wirtschaftsnahen Forschungseinrichtungen in Deutschland. In der internationalen Wissenschaftswelt zählt das DFKI zu den weltweit wichtigsten Centers of Excellence; ein zentrales Merk-mal der Arbeit des DFKIs ist die rasche Umsetzung von Spitzenforschung in praxisrelevante Anwendungslösungen. 1988 von namhaften deutschen Unternehmen der Informationstechnik und zwei Großforschungseinrichtungen als gemeinnützige GmbH gegründet, hat sich die DFKI GmbH inzwischen durch ihre proaktive und bedarfsorientierte Projektarbeit national und international den Ruf eines kompetenten und zuverlässigen Partners für Innovationen in der Wirtschaft erworben.

Im Forschungsbereich Sichere Kognitive Systeme am Standort Bremen unter der Leitung von Prof. Dr. Krieg-Brückner lag zum Zeitpunkt der Antragstellung der Schwerpunkt auf der Entwicklung sicherer und kognitiv adäquater technischer Systeme. Im Frühjahr 2012 wechselte die Leitung des Forschungsbereichs zu Prof. Rolf Drechsler, womit auch eine Neuausrichtung hinsichtlich einer formalen Systementwicklung und Umbenennung des Forschungsbereichs in Cyber-Physical Systems verbunden war. Die Arbeiten in SHIP wurden durch den Wechsel nicht beeinflusst.

Der Standort Bremen des DFKI verfügte aus mehreren einschlägigen Projekten über für SHIP relevante Expertise, insbesondere aus den folgenden Projekten: Das Projekt FORMALSAFE (For-

mal Development for Safe Robotics (BMBF, FKZ 01 IW 07002, 2007-2010) befasste sich mit dem integrierten Management formaler und halbformaler Dokumente sowie der Semantisierung sowohl von Dokumenteninhalten als auch von Metadaten. Ein Schwerpunkt dieses Projektes war insbesondere die Änderungsverwaltung der voneinander abhängigen Dokumente während ihrer Lebenszeit. Im Bereich der Ontologiesprachen leisteten die Projekte GENMOD bzw. GENMOD2 Generic Algorithmic Methods in Modal and Hybrid Logics (DFG, GZ SCHR 1118/5-1 und SCHR 1118/5-2, 2008-2013) wertvolle Vorarbeit, bei denen es um die Repräsentation natürlichsprachlichen Wissens und entsprechender darauf aufbauender Inferenzalgorithmen ging.

Das Projekt wurde von den folgenden Personen durchgeführt: Prof. Dr. Bernd Krieg-Brückner, Dr. Serge Autexier, Dr. Dieter Hutter, Dr. Christian Maeder, Dr. Dominik Dietrich (bis Februar 2013), Martin Ring (2013, Doktorand), MSc. Ewaryst Schulz (Doktorand) (until March 2012), Mathias Soeken (2013, Doktorand), Jannis Stoppe (2013, Doktorand).

Die folgenden Personen waren temporär unterstützend für das Projekt tätig: Dr. Christoph Lüth, Dr. Till Mossakowski (bis September 2013) Dr. Lutz Schröder (bis März 2012)

1.3 Planung und Ablauf des Vorhabens

Die Arbeit in SHIP gliederte sich zum einen in vier Arbeitsgebiete, die sich den Kernforschungsfragen im anvisierten Themenbereich widmen, einschließlich der Entwicklung der Werkzeugunterstützung in der SHIP-Umgebung für heterogene Entwicklung, Simulation und Monitoring von heterogenen Prozessen und Daten, und zum anderen die AAL-Fallstudie, in der die entwickelten Methoden und Werkzeuge zur Implementierung von altersgerechten Assistenzsystemen (AAL) eingesetzt werden sollten, und die Gegenstand eines eigenen Arbeitsgebiets AAL war (Arbeitsgebiet AAL). Diese Fallstudie lieferte die Szenarien, anhand derer die Methoden und Werkzeuge aus den Hauptarbeitsbereichen in SHIP evaluiert und demonstriert wurden.

Abbildung 1.3 zeigt eine Übersicht der Arbeitsbereiche und der zugehörigen Arbeitspakete in einem GANTT-Diagramm. Insbesondere gibt sie Auskunft über die zeitliche Planung der einzelnen Arbeitspakete. Diese ist eine Abstraktion der eigentlichen Ressourceneinteilung, bei der die vielen Querbezüge zwischen einzelnen Teilaufgaben zu einer teilweisen Parallelisierung der Aufgaben führte, die auch in der Abstraktion im PERT-Abhängigkeitsgraphen der Arbeitsschritte in Abbildung 1.2 nivelliert wurde. Insbesondere war Wissen aus der Fallstudie notwendig für die Entwicklung geeigneter Methodiken; andererseits waren existierende Methoden notwendig, um die Fallbeispiele überhaupt bearbeiten zu können. Insofern waren die zu den Arbeitspaketen angegeben Zeitintervalle die *spätest mögliche Intervalle*, in denen diese vollständig bearbeitet werden könnten. Die ursprünglich geplante Gesamtzahl der Personenmonate war 144. Im Sommer 2013 wurde das Projekt um 20 Personenmonate aufgestockt, um ein weiteres Anwendungsszenario *SHIP4AAL*. Dieses Teilprojekt wurde in Kooperation mit dem Robotic Innovation Center (RIC) des DFKI in Bremen durchgeführt, d.h. die Arbeiten wurden hälftig am RIC und im Forschungsbereich CPS durchgeführt. Tabelle 1.2 gibt eine Übersicht der Arbeitspakete und ihrer Aufteilung.

Kürzel		Thema	PM
FF		**Formales Rahmenwerk für heterogene Daten und Prozesse**	36
	FF-SL	Entwurf der Spezifikationsebene	16
	FF-Tool	Integration und Werkzeugunterstützung	10
	FF-Int	Generische Mechanismen zur Integration von Prozessen und Daten	10
RA		**Verfeinerung und Abstraktion heterogener Daten und Prozesse**	15
	RA-AR	Verfeinerung und Abstraktion	10
	RA-View	Sichten	5
VC		**Verifikation**	33
	VC-Mod	Verifikationsunterstützung für Koalgebraische Modallogiken	9
	VC-Pro	Werkzeugunterstützung für Heterogene Prozesse	15
	VC-Sim	Symbolische Auswertung durch Simulation	9
FO		**Flexible Orchestrierung von Daten und Prozessen**	24
	FO-Kon	Entwurf der Daten- und Prozesskonnektoren	12
	FO-Con	Konsistenzmanagement zur Laufzeit	12
AAL		**AAL Fallstudie**	36
	AAL-Req	Anforderungsspezifikation	4
	AAL-Spec	Modellierung und Verifikation	16
	AAL-Int	Anbindung von SHIP an das BAALL und AAL-Simulationswerkzeuge	16
S4A		**SHIP4AAL: Integration mit einer Robotikumgebung**	20
	S4A-IU	Intelligente Umgebungen	4
	S4A-SM	Semantische Modellierung	6
	S4A-TI	Technische Integration in das BAALL	4
	S4A-DZ	Demonstrationsszenarien	6

Tabelle 1.1 Übersicht über die Arbeitspakete in SHIP und SHIP4AAL

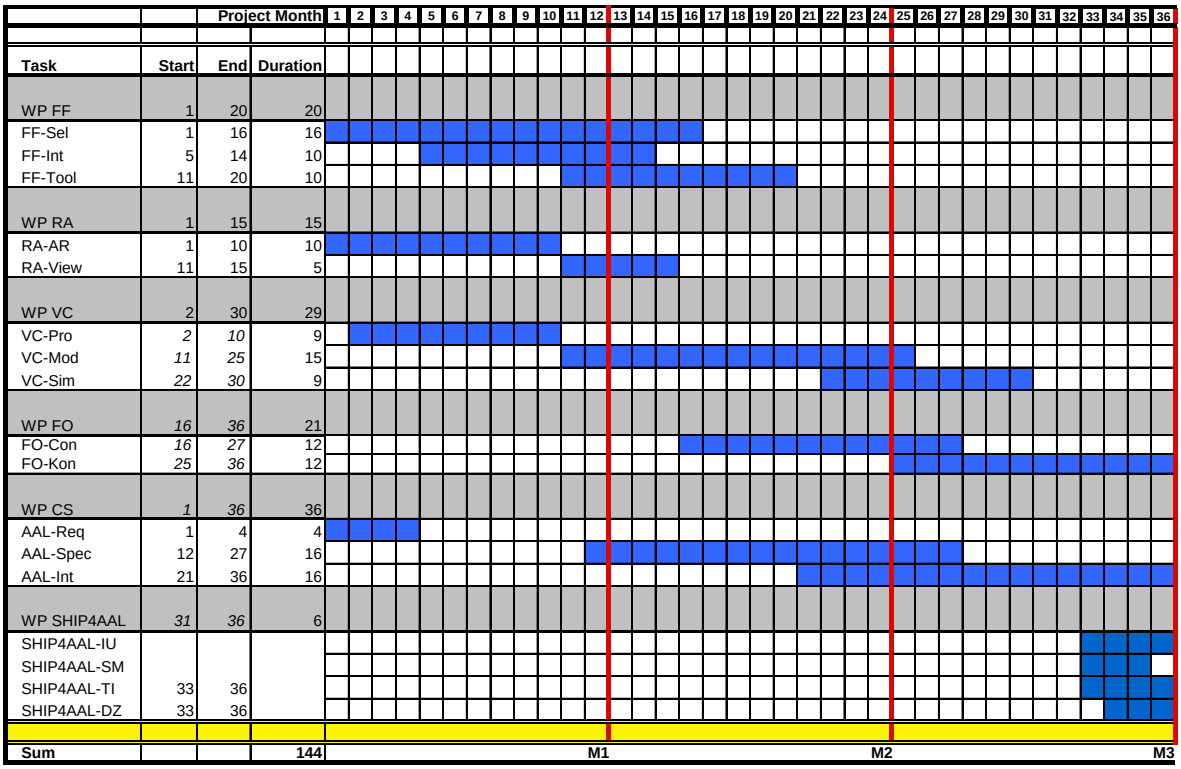

Tabelle 1.2 Zeitlicher Verlauf der Arbeitspakete in SHIP

1.4 Wissenschaftlicher und Technischer Stand zu Beginn des Vorhabens

1.4.1 Eigene Vorarbeiten

Heterogene Spezifikations- und Verifikations-Formalismen

Komplexe Syteme lassen sich oft nicht mehr innerhalb eines Formalismus beschreiben. Stattdessen werden mehrere Sichten (z.B. Daten, Prozesse, Struktur) auf das System entwickelt, deren Spezifikationen verschiedene Formalismen erfordern. Zudem können sich während einer formalen Entwicklung die verwendeten Formalismen ändern.

Während bisher existierende Ansätze integrierter formaler Methoden nur ad-hoc-Integrationen bestimmter Formalismen bereit stellen, bildet das Heterogeneous Tool Set Hets [MML07, CM08, CLMM09, Mos05a] einen allgemeinen Rahmen für die Integration formaler Methoden und für ein heterogenes Beweismanagement. Hets funktioniert dabei ähnlich wie ein Motherboard, das verschiedene Erweiterungskarten aufnehmen kann, solange die entsprechende Schnittstellenspezifikation eingehalten wird. Die Erweiterungskarten sind hier einzelne Formalismen und Logiken sowie Logik-Übersetzungen. Die Schnittstellenspezifikation basiert auf der Theorie einer Institution (einer Formalisierung des Begriffs des logischen Systems [GB92]); auf diese Weise wird eine semantisch fundierte Integration erreicht. Einzelne Logiken (zusammen mit ihren Analyse- und Beweiswerkzeugen) können über eine objekt-orientierte Haskell-Schnittstelle, die sich an Institutions orientiert, in das Hets-Motherboard eingesteckt werden.

Hets integriert bereits ca. 25 Logiken und zahlreiche Übersetzungen. Angebunden sind beispielsweise die SAT-Solver zChaff und miniSat, die automatischen Beweiser SPASS, E, Vampire, EKrHyper und Darwin, die interaktiven Beweiser VSE und Isabelle, der Modellfinder Darwin, das Rewriting-Tool Maude und viele andere mehr. Hets unterstützt die erwähnten 25 Logiken in ihrer jeweiligen Syntax. Für die Strukturierung von Spezifikationen unterstützt Hets zudem eine Reihe von Sprachen wie CASL, Common Logic, OWL-DL, Haskell und Maude. Für heterogene Spezifikation wurden die Strukturierungskonzepte von CASL, die besonders auf Betreiben unserer Bremer Forschungsgruppe bereits institutionsunabhängig formuliert wurden [BM04, CoF04, Mos00], zu der Sprache HetCASL (heterogenes CASL) erweitert. HetCASL erlaubt die heterogene Kombination von Spezifikation, die in verschiedenen Logiken geschrieben

wurden, mit Hilfe passender Logikübersetzungen.

Mit HETS wurde zum Beispiel der qualitativ räumliche Kalkül RCC8 verifiziert [WMS07]. Es wurde die Korrektheit der Kompositionstabelle von RCC8 in einem beliebigen topologischen Raum gezeigt. Die Axiomatisierung topologischer Räume erfolgte in Logik zweiter Stufe. Mittels einer Brückenspezifikation konnte die Verifikationsaufgabe in zwei Teile aufgeteilt werden: eine interaktive Verifikation weniger Beweisziele mittels Isabelle/HOL, und eine automatische Verifikation der meisten Beweisziele mittels SPASS.

Ein anderes Beispiel ist die mittels HETS erfolgte Verifikation der Konsistenz der Ontologie DOLCE [KLM08]. Die besten first-order-Modellfinder wie Darwin scheitern an der Komplexität von DOLCE. Mittels sogenannter Architekturspezifikationen konnte das Konsistenzproblem in kleinere Konsistenz- und Konservativitätsprobleme zerlegt werden. HETS selbst verifiziert automatisch dabei die Korrektheit der Zerlegung.

PROZESSSPEZIFIKATIONSSPRACHEN

CSP-CASL [Rog06, MR07, OMR12] kombiniert Spezifikationen erster Stufe mit der Prozessalgebra CSP. Dies ist u.a. im Rahmen der Verifikation eines elektronischen Zahlungssystems eingesetzt worden. Gängige Werkzeuge können die Verifikation von Sicherheitseigenschaften hier nur bis zu einem bestimmten Punkt unterstützen; insbesondere werden lose Anforderungsspezifikationen, die dann später zu einem Entwurf verfeinert werden können, nicht kombiniert für Daten *und* Prozesse unterstützt. Die dabei eingesetzte Technik der Integration von Daten und Prozessen ist im Prinzip auch für andere Prozessalgebren übertragbar; allerdings ist diese Generizität noch nicht ausgenutzt worden.

Die in Abschnitt 1.4.1 genauer diskutierten koalgebraischen Modallogiken dienen neben Zwecken der Wissensrepräsentation auch direkt als Spezifikationssprachen für Prozesse und reaktive Systeme. Dies gilt in besonderem Maße für probabilistische, gradierte und spieltheoretische Varianten, insbesondere insoweit hierbei sogenannte iterative Axiome (im klassischen Fall etwa Transitivität) oder Fixpunktlogiken mit abgedeckt werden, die eine über die bloße Spezifikation des jeweils nächsten Schrittes hinausgehende echt temporale Spezifikation der Langzeitentwicklung eines Systems erlauben. Resultate über iterative Axiome finden sich etwa in [PS08, SP08a]. Fixpunktlogiken werden generisch im Rahmen des jüngst entwickelten koalgebraischen μ-Kalküls erfasst; erste Komplexitätsresultate in diesem Bereich [CKP09] beruhen

entscheidend auf Vorarbeiten zu schnittfreien Regelmengen für Einschrittlogiken [SP09a]. Generische Resultate zur vollständigen Axiomatisierung und Tableaukalkülen für das Einvariablenfragment des koalgebraischen μ-Kalküls [SV10] liefern durch Anwendung im Einzelfall entsprechende neue Resultate z.B. für den gradierten μ-Kalkül [KSV02] und den alternierenden μ-Kalkül [AHK02].

Wissensrepräsentationssprachen

Im Zentrum des weitverzweigten Gebiets der Wissensrepräsentationssprachen steht heute die Familie der *Beschreibungslogiken* [BCM+03a], d.h. letztlich klassischer modaler bzw. hybrider Logiken mit relationaler Semantik (Kripke-Semantik) in leicht angepasster Syntax. Um dieses Zentrum herum gruppieren sich Logiken, die vom klassischen relationalen Paradigma abweichende Ausdrucksmittel bieten, etwa probabilistische, nichtmonotone, spieltheoretische oder agentenorientierte Logiken. Es gibt zunehmend Bemühungen, diese Ausdrucksmittel in Standardbeschreibungslogiken zu integrieren (z.B. [Luk08, LS08]). Hierzu haben die Antragsteller durch Arbeiten über spezifische logische Ausdrucksmittel [LS10, SPH10], über Anwendungen von Ontologiesprachen im für das aktuelle Projekt wichtigen Bereich der Ingenieurstechnik [FKS10] sowie durch die Entwicklung einer generischen Metatheorie der modalen und hybriden Logik jenseits der Kripke-Semantik, basierend auf Methoden der koalgebraischen Logik, beigetragen. Von besonderem Interesse sind im Zusammenhang mit diesem Projektantrag die in diesem Rahmen entwickelten generischen algorithmischen Methoden, deren Möglichkeiten zuerst durch die Antragsteller erkannt wurden [Sch07]. Zu nennen sind hier (neben ebenfalls für Wissensrepräsentationssprachen relevanten Ergebnissen, die in Abschnitt 1.4.1 bereits erwähnt wurden) etwa

- generische obere Schranken *PSPACE* und *NP* (mit tableau-orientierten und mit semantischen Methoden) für Deduktion über leeren TBoxen in modalen und hybriden Logiken [MPS09, SP08b, SP09b, PS10, PS09];
- eine generische obere Schranke *EXPTIME* für Deduktion in koalgebraischer Hybridlogik über allgemeinen TBoxen, d.h. letztlich für koalgebraische Beschreibungslogik [SPK09];
- ein globaler Caching-Algorithmus für effizientes Schließen in koalgebraischen Beschreibungslogiken [GKPS10]; und

- ein Modularitätsprinzip für Deduktion in heterogenen Modallogiken [SP07].

Hinzu kommen weitere metatheoretische Resultate, wie etwa starke Vollständigkeit modaler und hybrider Logiken [SP09b, SP10b] und ein Charakterisierungssatz vom van Benthem/Rosen-Typ zur modalen Ausdrucksstärke [SP10a]. Die oben genannten Algorithmen sind z.T. im Rahmen des generischen Deduktionswerkzeugs CoLoSS (Coalgebraic Logic Satisfiability Solver)[1] implementiert [CMPS09, HSar].

Änderungsmanagement

In dem früheren BMBF-Projekt PADS wurde von den Antragstellern der formale Begriff eines Entwicklungsgraphen [Hut00] als die logische Repräsentation einer konsistenten und strukturierten formalen Softwareentwicklung geprägt. Die Knoten dieses Entwicklungsgraphen entsprechen dabei verschiedenen Entitäten einer formaler Entwicklung (wie beispielsweise Theorien oder Module), während die Kanten gegebene oder postulierte Beziehungen zwischen den Entitäten repräsentieren (“erfüllt”, ”implementiert”, oder “verwendet”). Der Entwicklungsgraph fungiert dabei als strukturierte Datenbasis für verschiedene Verifikationswerkzeuge, die inkrementell während des Entwicklungsprozesses aufgebaut wird. Der Entwicklungsgraph erfüllt dabei zwei Aufgaben. Auf der einen Seite stellt er eine generische Repräsentationssprache für strukturierte Spezifikationen zur Verfügung. Auf der anderen Seite dient er als Truth-Maintenance-System auf der Ebene der Theorien, da er die Beziehungen zwischen Theorien über Änderungen hinweg verwaltet. Um eine solche Verwaltung von bestehenden Beziehungen zu ermöglichen, erlaubt der Entwicklungsgraph nur eine beschränkte Menge von Basisoperationen, deren Semantik dem System bekannt ist und es damit die Gültigkeit der bestehenden Abhängigkeiten und Beweise überprüfen kann [AH05].

Die Idee des Entwicklungsgraphen stellte sich als sehr fruchtbar für weitere Forschungsarbeiten heraus [HS01, MAH01, SH02, MAH06]. Zahlreiche Forschungsgruppen bauten auf diesem Ansatz auf. Till Mossakowski erweiterte beispielsweise die Basis der Entwicklungsgraphen hinsichtlich der Unterstützung verschiedener Logiken (heterogene Entwicklungsgraphen [MML07, Het, Mos05b]). Die Beweissemantik der Spezifikationssprache CASL (s. 1.4.1) wurde in Termini von Entwicklungsgraphen definiert [MHAH04]. Weiterhin wurden die Strukturierungsmittel von Entwicklungsgraphen in OMDoc [Koh06] übernommen, ein auf XML basiertes, inhaltsorientiertes Abbildungsformat für wissenschaftliche Dokumente. Schließ-

[1] http://www.informatik.uni-bremen.de/cofi/CoLoSS/

lich wurden Entwicklungsgraphen dazu verwendet, um die drei verschiedenen Kontexttypen in Isabelle, d.h. Theorien, Beweise und Locals, in uniformer Weise zu beschreiben. Entwicklungsgraphen wurden sowohl in dem MAYA-System [AHMS02] als auch in dem bereits oben erwähnten HETS-System (s. 1.4.1) verwendet, um die Entwicklung formaler, strukturierter Spezifikationen zu unterstützen.

Eine der Schlüsselbeobachtungen, die während der Entwicklung des MAYA-Systems gemacht wurden, war, dass die Mechanismen, die zur Realisierung des verteilten Änderungsmanagements zum Einsatz kamen, hauptsächlich auf den Strukturierungen der in Anwendungen vorkommenden Entitäten (und weniger auf der konkreten Semantik der behandelten Objekte) beruht [Hut04]. Um die steigende Komplexität bei realistischen Anwendungen zu beherrschen, wurden zahlreiche Typen von Dekompositionen für Beziehungen eingeführt, die dann ein hierarchisches Änderungsmanagement erlaubten.

In dem BMBF-Projekt MMISS [MMi] wurden erste Schritte unternommen, um diese Techniken auch auf die Verwaltung von Lehrmaterialien auszudehnen. Im Gegensatz zu formalen Entwicklungen haben diese Dokumente in der Regel keinen festen Konsistenzbegriff. Die zentrale Idee in diesem Projekt war, dem Entwickler eines Kurses die Möglichkeit zu geben, seine eigenen Konsistenzbedingungen in termini von Dokumentstruktur, Sprachvarianten, logische Abhängigkeiten, Versionen und Detaillierungsebenen zu formalisieren und damit das Änderungsmanagement zu steuern. Hierbei wurde der Begriff einer *semantischen Vernetzung von Dokumenten mittels einer Ontologie* durch Krieg-Brückner et al. [KBLL+04] entwickelt. Eine Domänenontologie wurde dabei parallel zu (möglicherweise mehreren) Dokumenten entwickelt und entsprechend strukturiert (siehe auch [MKB04]). Einträge in einer solchen Ontologie werden zur gegenseitigen Referenzierung und damit einer semantischen Bezugnahme verwendet.

In Zusammenarbeit mit NASA Ames wurden erste Schritte durchgeführt, um die Konzepte für das Änderungsmanagement in Formalen Methoden auch auf Programmzertifizierung und Managementsysteme zur Programmzertifizierung [DFHJ05] zu übertragen. Das Ziel ist ein System, mit dem Zertifikate gespeichert und wieder abgerufen werden können und im Fall von Änderungen Programme inkrementell wieder rezertifiziert werden können.

Diese Ansätze zur Verwendung von Strukturierungen für das Änderungsmanagement wurden im Vorgängerprojekt FormalSafe aufgegriffen, um ein allgemeines Rahmenwerk [Hut09] zu entwickeln, dass allgemein Konsistenzbegriffe für heterogene (nicht-formale) Dokumentkollektionen zu definieren erlaubt. Konsistenzbedingungen operieren dabei auf den Strukturierungs-

mechanismen der verschiedenen Dokumenttypen und setzten diese für einzelne Dokumentteile miteinander in Beziehung oder beschränken die Art und Weise, wie sie zusammengesetzt werden können. Die Verfolgung der verschiedenen Strukturen in den Dokumentkollektionen ermöglicht es, diejenigen Stellen eines Dokuments aufzuspüren, die von einer lokalen Änderung eines (anderen) Dokumentes betroffen sind, und sie gibt gleichzeitig Hinweise, wie diese Stellen an die Änderungen anzupassen sind, so dass die Konsistenz der Dokumentkollektion wieder gewährleistet ist.

Diese Ideen wurden konkret in dem Werkzeug GMoC [AM10] umgesetzt, das semantische Analyse und Vernetzung von Dokumenten in einer Dokumentenkollektion und dabei insbesondere die semantische Analyse von Änderungen und ihrer Auswirkungen auf die gesamte Dokumentenmenge unterstützt. GMoC ist als Ergänzung zu den Änderungsmanagement-Funktionalitäten in existierenden Autorierungs-Werkzeugen bzw. integrierten Entwicklungsumgebungen für spezielle Dokumententypen angelegt, die selbst keine Unterstützung für die Auswirkung von Änderungen auf andere Dokumentenformate bieten. Aufbauend auf deklarativen Spezifikationen der semantischen Strukturen spezifischer Dokumenttypen in sogenannten Dokumentenmodellen, können in Interaktionsmodellen deklarative Propagationsregeln definiert werden, die die Auswirkungen semantisch-relevanter Änderungen auf andere Dokumente operational beschreiben. Die Dokumentenmodelle und Interaktionsmodelle bilden die Parameter für das GMoC-Werkzeug zur Bereitstellung von semantischer Analyse and Änderungsmanagement für heterogene Kollektionen von Dokumenten.

Die grundlegende Idee von GMoC ist es, alle Dokumente und ihre intendierte Semantik in dem gleichen strukturierten Graphen gemeinsam zu repräsentieren, um Konsistenzprüfungen durchzuführen und die Auswirkungen der Änderungen berechnen zu können. Die Berechnung der semantischen Informationen, die Konsistenzprüfungen und die Auswirkungen der Änderungen werden mit Hilfe von Graphersetzungsregeln operationalisiert, die in den Dokumentenmodellen bzw. Interaktionsmodellen definiert werden. Hierzu wurde in GMoC das Graphersetzungswerkzeug GrGen [GBG+06] als Teilwerkzeug integriert. Zur Berechnung von semantisch-relevanten Änderungen bzw. zur Ausblendung semantisch-irrelevanter Änderungen zwischen zwei Versionen einer Dokumentenkollektion wird eine Differenzanalyse zwischen den verschiedenen Dokumenten verwendet, die wiederum parametrisiert über Dokumenttyp-spezifische semantische Äquivalenzmodelle ist.

1.4.2 Fachliteratur und verwendete Dokumentationsdienste

Die verwendete Fachliteratur ist im Literaturverzeichnis ab Seite 57 angegeben.

1.5 Zusammenarbeit mit anderen Stellen

SHIP kooperierte insbesondere mit den folgenden Projekten in anderen Forschungsbereichen des DFKIs bzw. an der Universität Bremen:

- DFKI - Intelligente Benutzerschnittstellen: Modellierung der Domänen in Ontologien, Komposition von Services
- Prof. Dr. Carsten Lutz: Probabilistische Ontologien
- Transregionaler Sonderforschungsbereich SFB/TR8 in Bremen und Freiburg zum Thema "Raumkognition": Orientierung und Lokalisation in instrumentierten Umgebungen BAALL.

2 Eingehende Darstellung (NKBF 98 8.2 II)

Das Ziel des Projekts SHIP war es, eine formale Methodik und eine Werkzeugunterstützung zur Entwicklung zuverlässiger, verteilter und heterogener Prozesse zu entwickeln. Ausgehend von einer Anforderungsspezifikation der Eigenschaften eines Gesamtsystems sollte deren Verfeinerung unter Einbeziehung existierender Komponenten bis hin zu einer Implementierungsebene unterstützt werden. Weiterhin sollten exemplarisch Assistenzprozesse für das Bremen Ambient Assisted Living Lab (BAALL) in dieser Methodik realisiert werden, um die dort existierenden Soft- und Hardwarekomponenten zu integrieren.

Ein Ergebnis von SHIP ist ein reichhaltiger Spezifikationsformalismus, der zum einen Modallogik und Temporallogik integriert und zum anderen einen einheitlichen Formalismus bietet, um eigenschaftsorientiert komplexe Assistenzprozesse zu spezifizieren. Diese Prozesse setzen sich aus aktiven Prozesstätigkeiten und aus passiv die Entwicklung der Umgebung beobachtenden Wartephasen zusammen. Dadurch können Prozesse sowohl abstrakt als auch implementierungsnah spezifiziert werden. Auf der Implementierungsebene wird der Zustand eines Systems und seiner Umgebung in einer Beschreibungslogik (OWL) modelliert. Änderungen in der Umgebung werden durch entsprechende Änderungen des logik-basierten Modells nachvollzogen. Umgekehrt, werden in dem Modell initiierte Änderungen durch entsprechende Aktoren (technische Systeme) in der Umgebung auf die reale Welt übertragen. Die zugrunde liegende Prozesssprache ist eine skriptähnliche Programmiersprache, die als Basisoperationen zum einen Aktionen, die das semantische Modell verändern, und Monitore, die auf ein in Temporallogik spezifizierte Entwicklung der Umgebung warten, zur Verfügung stellt. Das im Projekt entwickelte SHIP-TOOL beinhaltet einen Interpreter für diese Sprache sowie Schnittstellen zur bidirektionalen Synchronisation von Umgebung und Komponenten mit deren semantischen Modell. Dadurch können entwickelte Prozesse sowohl zur Simulation als auch zur Interaktion mit der realen Welt verwendet werden.

Das SHIP-TOOL wurde erfolgreich an mehreren Fallstudien evaluiert. Zum einen wurde für eine intelligente Orchestrierung der verschiedenen autonomen Prozesse im BAALL verwendet. Zum

andern wurde das SHIP-TOOL zur Unterstützung der Einhaltung klinischer Leitlinien, indem die durch die Leitlinien vorgeschlagenen Entwicklungsprozesse durch entsprechende Monitore spezifiziert wurden.

Die folgenden Abschnitte beschreiben eingehend die vorgenommenen Arbeiten und Untersuchungen im Vorhaben SHIP sowie deren Ergebnisse.

2.1 Verwendung der Zuwendung

2.1.1 Arbeitspaket FF: Formales Rahmenwerk für heterogene Daten und Prozesse

Das Projekt begann 2011 mit der Auswahl und dem Design einer geeigneten Spezifikationssprache für das zu entwickelnde SHIP-Rahmenwerk. Dieses Design wurde maßgeblich von den in Fallstudien BAALL und SmartTies auftretenden Typen von Anforderungen bestimmt (vgl. hierzu den Abschnitt 2.1.5 auf Seite 39). Ausgangspunkt waren die Anforderungen, beispielsweise dynamisch Prozesse zu erzeugen und zu beenden, Prozesse auf bestimmte Entwicklungen der Umwelt warten und reagieren zu lassen, oder allgemein verschiedene parallele Prozesse mit externen Systemen oder Benutzern in der realen Welt zu verknüpfen.

Diese Arbeiten mündeten in einer Logik ExtModal, die die Spezifikation abstrakter Prozesse erlaubt, und einer logik-basierten Programmsprache für das SHIP-TOOL, das als Grundlage für die Simulation und Implementierung der Systeme dient. Auf beides wird im Nachfolgenden genauer eingegangen. Abbildung 2.1 gibt einen Überblick über die verschiedenen Spezifikationsebenen in SHIP.

ABSTRAKTE SPEZIFIKATIONSEBENE – EXTMODAL

Ausgangspunkt für die Entwicklung einer geeigneten Spezifikationssprache für höhere Abstraktionsebenen war zunächst CSP-CASL. Da Spezifikationen in CSP-CASL eine feste Anzahl von Prozessen erzwingen, konnte diese Sprache nicht in SHIP eingesetzt werden und es mussten entsprechende Weiterentwicklungen vorgenommen werden. Dabei ergaben sich

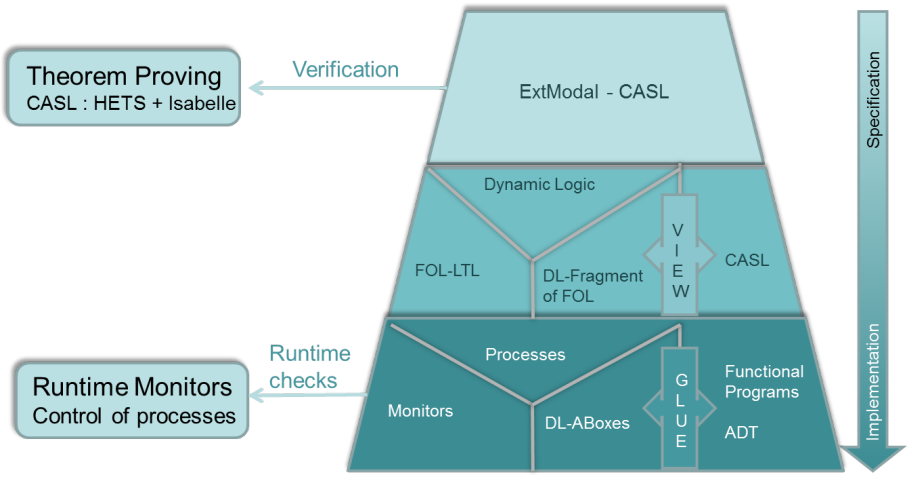

Abbildung 2.1 Hierarchische Spezifikationsmechanismen in SHIP

bei der Analyse von Prozessen mit reichhaltigen Datenstrukturen zwei wichtige Beobachtungen: Zum einen ermöglichen es Prozessspezifikationen, die die Verwendung reichhaltiger Datentypen erlauben, Workflowbeschreibungen selbst als einen Datentyp aufzufassen, womit Prozesse zu (generischen) Interpretern dieser Beschreibungen werden. Dies erlaubt es, Flexibilität und Anpassbarkeit von Workflows zu untersuchen, indem Eigenschaften einer Workfloworchestrierung oder Reorganisation bereits auf der Spezifikationsebene formalisiert und verifiziert werden. Zum anderen können Systemzustände solcher Prozesse als Daten interpretiert werden, womit man einen fließenden Übergang von aktionsorientierten Spezifikationen von Prozessen (wie z.B. in Prozesskalkülen) hin zur Spezifikation von temporären Eigenschaften in der Evolution der Zustandsvariablen erreicht.

Hierfür wurden zunächst die Grundlagen für die auf koalgebraischen Modallogiken basierenden Spezifikationsmechanismen, wie sie für SHIP benötigt wurden, entwickelt. Insbesondere wurden Aspekte der quantitativen Unsicherheit [GBJLS11a, GBJLS11b] und der vagen Konzepte [SP11a] analysiert. Das dabei entstandene Manifesto der Koalgebraischen Logik wurde in dem angesehenen "Computer Journal"publiziert [CKP+11].

In dem für die Spezifikation und Verifikation vorgesehene Hets-System war bereits zu Beginn des Projektes eine Logik realisiert, die eine Multimodal-Logik mit einer Branching-Time-Logik kombiniert. Im Rahmen von SHIP wurde diese Logik um eine dynamische Logik für parallele Prozesse erweitert, wobei Programmausdrücke als Termmodalitäten über definierbare Algebraische Datentypen und den üblichen Programmkonstruktoren für sequentielle Komposition, Tests, Alternativen und parallele Ausführung definiert werden. Parallele Ausführung wird

mit Hilfe einer Interleaving-Step-Semantik interpretiert, dabei können Prozesse durch Parameter über algebraische Datentypen benannt werden. Das Verhalten der Programme wird über Frame-Axiome definiert. Die sich daraus ergebende Logik **ExtModal** erlaubt zum einen die Formalisierung paralleler, rekursiver Prozesse und zum anderen die Angabe temporallogischer Formeln zur Beschreibung des erwarteten Verhaltens externer Komponenten oder der Eigenschaften des kombinierten Systems.

Zur Illustration betrachten wir im Nachfolgenden ein einfaches Türkontrollprogramm, dass in dem Anwendungsszenario BAALL sicher stellt, dass nachts alle Türen geschlossen sind und nur im Notfall automatisch geöffnet werden und bis zum Ende des Notfalls offen bleiben. Der Datentyp zur Spezifikation dieses Programmes in Dynamischer Logik sieht wie folgt aus:

```
free type Step ::= LockDoor(Door) %% Basic Program step
                 | UnlockDoor(Door) %% Basic Program step
               | NightComes(Time) %% Monitoring step
               | EmergencyMonitor(Situation,Door) %% Monitoring Step
             | EmergencyHappens(Situation,Time) %% Monitoring Step
ops DoorControl : Time * Door -> Step;
    EmergencyOpen : Situation * Door * Time -> Step;
    System : Situation * Time * Door -> Step;
```

Das Verhalten des Programmschritts zum Schließen der Tür ist durch das folgende Frame-Axiom spezifiziert.

```
var d:Door . d = Open => [LockDoor(d)] d = Locked %% LockDoor
```

Dieses formalisiert, dass eine offene Tür d in allen erreichbaren Welten nach Durchführung der Aktion $LockDoor(d)$ geschlossen sein wird. Monitorformeln werden zur Spezifikation des erwarteten Verhaltens der externen Systeme, der Benutzer oder der realen Welt verwendet. Zum Beispiel kann der Monitor, der so lange wartet bis die Nacht herein bricht, durch das folgende Frameaxiom beschrieben werden:

```
forall t:Time . [NightComes(t)] PHI <=> t = Day U (t = Night /\ PHI) %% ↩
    NightComes Monitor
```

dabei ist `PHI` eine Variable zweiter Ordnung und `U` der Until-Operator der verwendeten Temporallogik. Komplexe Programme für die Türkontrolle können damit durch die folgenden Frameaxiome beschrieben werden:

```
forall t:Time; d:Door .
```

```
[DoorControl(t,d)] PHI <=>
[ (t = Day)? ; NightComes(t) ; DoorControl(t,d)
  orElse (t=Night /\ d=Locked?)
  orElse t=Night /\ not d=Locked? ; LockDoor(d)] PHI ;
```

dabei ist *PHI* wie zuvor und spezifiziert, dass das Verhalten der Türkontrolle für eine Tür *d* zu einem Zeitpunkt *t* durch eine alternierende Folge der folgenden Aktivitäten definiert ist: (1) falls es Tag ist, wird mit Hilfe der entprechenden Monitorformel auf die Nacht gewartet und anschließend rekursiv aufgerufen, (2) falls es Nacht ist und die Tür geschlossen ist, wird die Türkontrolle beendet, oder (3) falls es Nacht ist und die Tür geöffnet ist, wird die Tür geschlossen und der Prozess beendet. Das komplette System ist darauf aufbauend folgendermaßen spezifiziert:

```
forall  sit:Situation ; t:Time ; d:Door .
   [System(sit,t,d)] PHI <=> [DoorControl(t,d) ; EmergencyOpen(sit,d,t)] PHI
```

wobei der Prozess, der auf Notfallsituationen reagiert, dann aufgerufen wird, wenn der Preozess der Türkontrolle beendet ist. Dieser ist wie folgt definiert:

```
forall sit:Situation; t:Time ; d:Door .
  [EmergencyOpen(sit,d,t)] PHI <=>
  [((d=Locked? ; UnlockDoor(d) ; EmergencyMonitor(sit,d))
   orElse (d=Open? ; EmergencyMonitor(sit,d))) ; System(sit,t,d)] PHI ;
```

Dieser Prozess wartet mittels eines Monitors auf eine Notfallsituation, öffnet im Notfall die Tür, falls diese geschlossen sein sollte, und wartet auf das Ende des Notfalls. Danach ruft sie rekursiv den gesamten Prozess wieder auf.

Trotz seiner Einfachheit, demonstriert dieses Beispiel anschaulich wie die Spezifikation von aktiven Prozessen und Monitore, die die Umwelt beobachten, in einem einheitlichen Formalismus erfolgen kann.

ExtModal wurde in das Hets-System als eine Erweiterung der CASL-Logik integriert und kann damit zur Spezifikation von Prozessen und Monitoren unter Verwendung der reichhaltigen abstrakten Datentypen von CASL verwendet werden.

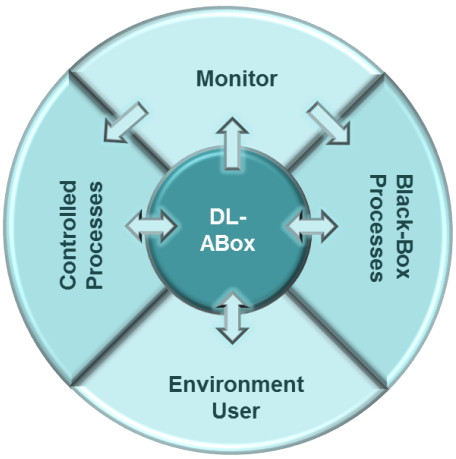

Abbildung 2.2 Interaktion in SHIP

Implementierungsnahe Ebene – Dynamische Logik und Temporallogik

Für die implementierungsnahe Spezifikation von Prozessen und ihrer Interaktionen wurde eine logik-basierte Programmiersprache entwickelt, die eine (dynamische) Deskriptionslogik mit einer linearen Temporallogik verbindet. Sie erlaubt nicht nur die Definition von Prozessen sondern ermöglicht auch bereits auf abstrakter Ebene erwähnte Definition von Monitoren, die Entwicklungen ihrer Umgebung beobachten und entsprechend auf diese reagieren können. Technisch erfolgt die Spezifikation solcher Monitore als Formeln in linearer Temporallogik, die die Anforderungen an die temporale Entwicklung der Umgebung formal repräsentieren. Mit Hilfe der sogenannten Formelprogression werden diese Anforderungen in die Zukunft (d.h. bei jedem sogenannten Update) fortgeschrieben.

Abbildung 2.2 illustriert das Gesamtbild, das der Entwicklung der Spezifikationssprache für implementierungsnahe Ebenen zu Grunde liegt. Im Zentrum steht ein globaler Zustand, der den aktuellen Stand des Gesamtsystems widerspiegelt. Dieser Zustand wird als ABox in einer Deskriptionslogik formalisiert. Die ABox wird ergänzt durch das in der TBox spezifizierte allgemeine Wissen. Zustandsänderungen erfolgen durch sogenannte ABox-Updates, in denen neue Fakten in die ABox eingebracht und dazu in Widerspruch stehende alte Fakten aus

der ABox entfernt werden. Die ABox ist typischerweise ein abstraktes Modell der realen Welt und Änderungen in der realen Welt werden über Updates der ABox in dem abstrakten Modell nachvollzogen. Updates können entweder durch externe Änderungen der realen Welt, die durch externe Systeme oder dem Benutzer verursacht werden, initiiert werden oder intern durch Prozesse, die auf der ABox selbst arbeiten, berechnet und anschließend auf die reale Welt übertragen werden. Diese Prozesse werden in einer dynamischer Deskriptionslogik spezifiziert; sie sind quasi Programme über Updates, die das elementare Programmkonstrukt dieser Programmiersprache bilden.

Aktionen definieren Updates mit Hilfe von Vorbedingungen und Effekten. Eine elementare Aktion arbeitet auf der aktuellen ABox und ist anwendbar, wenn die Vorbedingung in dieser ABox ableitbar ist. Die Aktion schlägt fehl, wenn die Vorbedingung falsifizierbar ist und sie wartet, falls keine der beiden erwähnten Situationen zutrifft.

Betrachtet man beispielsweise die Türsteuerung aus dem vorangegangenen Abschnitt, so wird die entsprechende Aktion zum Schließen der Tür wie folgt definiert:

```
action LockDoor(d) = {
 pre = d:Door, d:Open
 effect = d:Locked
}
```

Falls die Aktion anwendbar ist, bildet die neue Information $d : Locked$ gleichzeitig die technische Anforderung an den automatischen Türschließer der Tür d in der realen Welt. Parallelkomposition wird als Verschränkung der Einzelprozesse betrachtet (analog zu ExtModal). Dies bedeutet, dass alle Prozesse auf einer gemeinsamen ABox operieren, wobei zu jedem Zeitpunkt nur maximal eine Aktion eines dieser Prozesse auf der ABox ausgeführt werden kann. Bedingungen für Konditionale werden als Anfragen an die aktuelle ABox interpretiert; sie können erfüllt sein oder fehlschlagen.

Monitorformeln werden in einer linearen Temporallogik erster Ordnung über Fakten der Abox definiert. Quantifizierungen werden über die in der ABox verfügbaren Individuen durchgeführt. Beispielsweise wird der Monitor, der über den Tag/Nacht-Wechsel wacht, wie folgt definiert:

```
monitor NightComes(time) = time:Day U time:Night
```

Wird dieser Monitor gestartet, erfolgt eine Auswertung der Monitorformel anhand der in der ABox vorgenommenen Zustandsänderungen (Updates). Technisch erfolgt dies durch

sogenannte Formelprogression (siehe [BF12]), wie sie bereits aus der Laufzeitverifikation bekannt ist. ABox-Aussagen, die in dem Monitor vorkommen, bilden dabei Anfragen an die aktuelle ABox.

Betrachtet man beispielsweise eine Instanz von `NightComes(t)` bezüglich einer ABox, in der `t:Day` gilt, so resultiert die Formelprogression in der Formel `t:Day U t:Night`. Diese neue Formel wird bezüglich des Nachfolgezustands (ABox) ausgewertet und bleibt solange unverändert bis entweder `t:Night` oder **not** `t:Day` eintritt. Im ersten Fall evaluiert der Monitor zu **true**, womit der Monitor das spezifizierte Verhalten beobachtet hat. Im zweiten Fall evaluiert der Monitor zu **false**; dies bedeutet, dass das beobachtete Verhalten von der Spezifikation abgewichen ist.

2.1.2 Arbeitspaket RA : Verfeinerung und Abstraktion heterogener Daten und Prozesse

Dieses Arbeitspaket beschäftigte sich mit der Adaptierung bestehender Verfeinerungsbegriffe für Daten und Prozesse für die Bildung einer einheitlichen Verfeinerungssprache für heterogene Spezifikationen. Einen Schwerpunkt bildete dabei die Übertragung des Kompositionsbegriffs für Prozesskalküle, um damit die Verfeinerung abstrakter Eigenschaften kombinierter Systeme auf die Komposition abstrakter Eigenschaften in den Unterkomponenten zu ermöglichen. Weiterhin galt es den Vorteil einer reichhaltigen Datenrepräsentation auszunutzen, der es erlaubt, die Modellierung eines Prozesses in eine als abstrakter Datentyp formulierte Workflowbeschreibung und einen Interpreter für solche Beschreibungen aufzuspalten. Damit können sowohl Teile des Interpreters selbst wieder als Workflowbeschreibungen als auch Workflowbeschreibungen als Datentyp für maschinennähere Interpreter verfeinert werden. Dieser Ansatz bietet eine präzises Instrumentarium, um die gewünschten Aspekte eines Prozesses, die verfeinert werden sollen, selektieren zu können.

Die erweiterte Modallogik stellt dabei den Kompositionsformalismus, der aus Prozesskalkülen bekannt ist, zur Verfügung. Durch die Integration in das Hets-System stehen auch die Strukturierungsmittel für formale Spezifikationen aus Hets zur Verfügung. Weiterhin erlaubt es die Verfeinerung abstrakter Spezifikationen in konkretere Spezifikationen über den Theorieinklusionsmechanismus aus Hets. Dies ermöglicht sowohl einen Prozess in mehrere Einzelprozesse oder die Daten eines Prozesses flexibel in Hets zu verfeinern. Gleichzeitig erlaubt es auch, eine globale Systemspezifikation in systemeigene Prozesse und in Monitore,

die das Systemverhalten beobachten und ggf. darauf reagieren, zu zerlegen.

Diese reichhaltige Logik erlaubt damit, SHIP-Prozesse sowohl auf einem sehr hohen Abstraktionsniveau unter Verwendung reichhaltiger Datentypen als auch implementierungsnah mit Hilfe dynamischer Logik zur Formalisierung von Prozessen und Temporallogik für Monitorformeln zu spezifizieren. Alle diese Verfeinerungen und Kompositionen sind in derselben Logik ausdrückbar und es werden dadurch die üblichen Probleme vermieden, die sonst die Verwendung verschiedener Logikformalismen für verschiedene Ebenen auftreten könnten. Es zeigte sich allerdings auch, dass diese Uniformität der Spezifikation zu Lasten einer Automatisierung bei der Verifikation einher geht (vgl. Abschnitt 2.1.3).

2.1.3 Arbeitspaket VC: Verifikation

Unterstützung höherer Abstraktionsebenen

Die Verifikation der reichhaltigen Prozessspezifikationen und deren Verfeinerungen benötigen typischerweise Beweiser höherer Ordnung (z.B. Induktion). Aus diesem Grund wurden Beweiser höherer Ordnung in das Hets-System[1] integriert. Dazu wurde ein Logikknoten THF in den Hets-Logikgraph eingefügt, der den THF-Standard der TPTP[2]-Bibliothek für Logiken höherer Ordnung zur Verfügung stellt. Über diesen Knoten kann nun das Beweissystem LEO-II [BTPF08] angesprochen werden. Durch die Definition der entsprechenden Logikkomorphismen für die oben genannten Modal- und Temporallogik können jetzt Probleme, die in diesen Logiken spezifiziert werden, mit LEO II automatisch behandelt werden [BP08].

Die im Arbeitspaket FF durchgeführten Arbeiten an einer koalgebraischen Logik betrafen auch entsprechende Verifikationsmechanismen. Da koalgebraische Verfahren eher bei der Spezifikation einzelner Prozesse als bei der Spezifikation kombinierter Assistenzsysteme zum Einsatz kommen, sollte eine solche Logik entscheidbar sein. Daher wurden die Spezifikationsmechanismen für Datentypen im Vergleich zu CASL auf entscheidbare Constraints und feste (nicht-flexible) Prädikate eingeschränkt, womit es uns möglich ist, Dynamische Logik mit konkreten Domänen [BCM+03b] zu verwenden. Die Beweisunterstützung erfolgt durch das in Hets integrierte Beweissystem CoLoSS [CMPS09], das Erfüllbarkeitsprobleme in koalgebraischer Logik lösen kann.

[1] http://www.informatik.uni-bremen.de/agbkb/forschung/formal_methods/CoFI/Hets/
[2] http://www.tptp.org/

Weiterhin wurde ein abstraktes Rahmenwerk für parallele Prozesse entwickelt, das generische Seiteneffekte bei der Ausführung atomarer (Prozess-)Schritte erlaubt, die mit Hilfe des Monadischen Kapselungsprinzip für Effekte behandelt werden. In diesem Rahmenwerk werden Prozesse als potentiell unendliche Folge von Fortführungen betrachtet, die durch finale Koalgebren über einer monadischen Basis modelliert werden. Als Kalkül für diese Prozesse wurde eine Erweiterung der Moggi'schen monadischen Metasprache für Effekte entwickelt. Es konnte die Korrektheit und Vollständigkeit einer natürlichen Gleichheitsaxiomatisierung nachgewiesen werden. Weiterhin wurde ein Korekursionsschema bereitgestellt, das über die Basissprache explizit definierbar ist und ein flexibles Ausdrucksmittel bereitstellt, um neue Operatoren auf Prozessen, wie beispielsweise die parallele Komposition, definieren zu können. Als Anwendungsbeispiel wurde damit die Sicherheit eines generischen wechselseitigen Ausschlussschemas mit Hilfe einer Verifikationslogik nachgewiesen, die auf dem Gleichheitskalkül aufbaut.

Für die Verifikation spezifizierter Systeme wurde ein Komorphismus für die Übersetzung von ExtModal-Spezifikationen in die Logik höherer Stufe HasCASL in Hets realisiert. Damit können Spezifikationen in ExtModal in die Eingabesprache interaktiver Beweiser, wie beispielsweise Isabelle/HOL, übersetzt werden und dort auch mit Hilfe automatisierter Beweisverfahren verifiziert werden. Es ist allerdings anzumerken, dass diese Transformation die Lesbarkeit (und damit auch die Komplexität) der Spezifikationen deutlich verschlechtert, so dass eine Hilfestellung bei der Verifikation durch den Benutzer erschwert wird. Es ist daher notwendig, weitere Heuristiken oder Taktiken für die Verifikation dieser Aufgaben in diesen Beweisern zur Verfügung zu stellen, um den Automatisierungsgrad zu verbessern. Dies ist allerdings nicht Teil dieses Projektes gewesen.

Implementierung - SHIP-tool

Im Frühjahr 2012 wurde mit der Entwicklung des SHIP-tool begonnen, das es erlauben sollte, heterogene Prozesse und Monitore parallel und verschränkt auszuführen. Das grundlegende Konzept war, dass die Repräsentation der realen Welt abstrakt in der ABox einer OWL-Ontologie erfolgt. Änderungen in Welt werden in Updates der ABox übersetzt und umgekehrt. Prozesse setzen sich aus elementaren Aktionen mit Hilfe gängigen Programmkompositionskonstrukten, wie sie auch in ExtModal zur Verfügung stehen, zusammen.

Das Pellet-Beweissystem für OWL dient als vollständiges Inferenzsystem zur Beantwortung

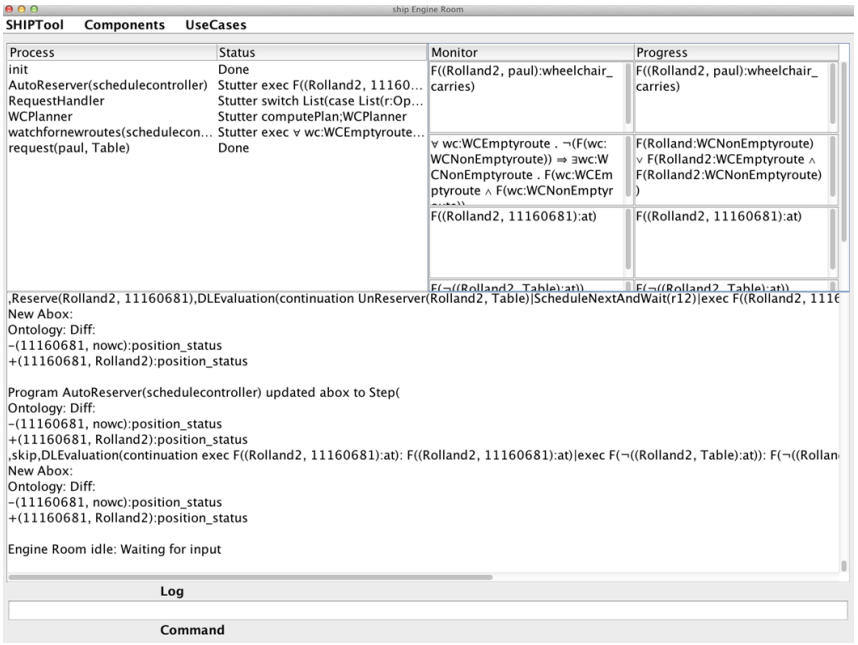

Abbildung 2.3 Benutzeroberfläche des SHIP-TOOL

von Anfragen an die Ontologie. Pellet unterstützt OWL2 und Query Answering. Monitore können in Prozessen aufgerufen werden. Ein Prozesse warten dann, bis der aufgerufene Monitor beendet ist. Der synchronisierte Aufruf eines Monitors erlaubt dem Prozess auf spezifizierte Entwicklungen der Umwelt zu warten und entsprechend darauf zu reagieren. Dieser Ansatz erweist sich damit als wesentlich flexibler als vergleichbare Ansätze (z.B. AgentSpeak), bei denen nur auf einzelne Events (aber keine längeren Entwicklungen) reagiert werden kann.

Das SHIP-TOOL erlaubt das Einlesen einer Hintergrundontologie, der ABox, sowie die Prozess- und Monitorspezifikationen. Der Benutzer kann manuell Prozesse und Monitore starten und beliebige Updates in das System einspeisen. Dies erlaubt insbesondere die Simulation von Prozessen und Monitoren. Da die Definition der Prozesse und Monitore ähnlich zu ihrer Spezifikation in ExtModal erfolgt, erwarten wir, dass es in Zukunft möglich sein wird, diese automatisch aus der untersten Spezifikationsebene von ExtModal zu erzeugen, um sie im SHIP-TOOL laufen zu lassen. Zusätzlich ermöglicht die Verwendung von Updates als Schnittstelle zwischen der realen Welt und den internen Prozessen es, das SHIP-TOOL direkt mit Hilfe von Monitoren mit externen Systemen, wie beispielsweise BAALL oder dem SmartTies System, zu koppeln.

Die Möglichkeit, in dem SHIP-TOOL Prozesse und Monitore eng verschränkt miteinander ablaufen zu lassen, erwies sich als eine geschickte Option, eine Laufzeitverifikation interner und

externer Prozesse durchzuführen. Eine schwerwiegendes Problem bei der Verifikation solcher Systeme besteht ja darin, dass das Verhalten der Umwelt nicht präzise genug erfassen oder einschränken lässt. Die SHIP-TOOL-Sprache erlaubt es dagegen, Assistenzprozesse zu spezifizieren, die parallel mit einem temporallogischen Monitor für das angenommene Verhalten der Umgebung ausgeführt werden. Falls die Umgebung diese Annahmen nicht erfüllt, schlägt der Überwachungsmonitor fehl und kann dadurch einen Prozess initiieren, der auf diese Situation aktiv reagiert.

INFORMATIONSSICHERHEIT

In vielen möglichen Anwendungsgebieten des SHIP-TOOL sind Datenschutzinteressen betroffen. Die Verwendung einer zentralen ABox zur Repräsentation der realen Welt sowie die Verwendung von Monitoren, die diesen globalen Speicher beobachten können, weckt weitere Sicherheitsbedenken. Es wurde daher ein Sicherheitskonzept für das SHIP-TOOL auf Basis einer Information-Fluss-Kontrolle realisiert. Die Idee ist, alle in der ABox befindlichen Fakten einzelnen Sicherheitsdomänen zuzuordnen. Eine Sicherheitspolitik spezifiziert, welche Domänen aus anderen Domänen einsehbar sind. Diese Sichtbarkeit muss mit dem Inferenzsystem insofern verträglich sein, als es nicht der Fall sein darf, dass aus sichtbaren Fakten auf nicht-sichtbaren Fakten geschlossen werden kann. Ein Benutzer oder Prozess ist selbst Mitglied einer Domäne und kann dementsprechend nur Fakten sehen, die in der Domäne sichtbar sind. Korrespondierend zu dem klassischen Bell/LaPadula-Ansatz für Vertraulichkeit kann er nur Fakten ändern, die nicht in unterliegenden Domänen sichtbar sind.

Die Umsetzung dieses Konzeptes erzwingt entsprechende Eingriffe in die Semantik des SHIP-TOOL. Insbesondere werden die deduktiven Fähigkeiten der Prozesse oder Benutzer auf die Fakten eingeschränkt, die in ihrer Domäne sichtbar sind. Konstrukte für die Allquantifizierung werden nur über die in einer Domäne sichtbaren Individuen interpretiert.

Es wurde gezeigt, dass ein solchermaßen konditionierter Ansatz die Vertraulichkeit nicht-sichtbarer Fakten in einer Domäne respektiert. Dies bedeutet, dass das sichtbare Verhalten des Systems für einen Beobachter, der in einer Domäne *D* lebt, nicht von Fakten oder Änderung von Fakten in nicht-sichtbaren Domänen abhängt. Dies gilt auch bei Einsatz von Monitoren, die ja bei *jeder* Änderung des globalen Zustands fortgeschrieben werden. Hier kann gezeigt werden, dass Änderungen im nicht-sichtbaren Bereich keine Veränderung der sichtbaren Monitorformeln bewirken.

2.1.4 Arbeitspaket FO: Flexible Orchestrierung von Daten und Prozessen

Das SHIP-TOOL kommuniziert ausschließlich über ABox-Updates mit seiner Umgebung (vgl. Abschnitt 2.1.4). Damit besteht die Schnittstelle des SHIP-TOOL aus verschiedenen Komponenten, die das SHIP-TOOL mit den entsprechenden externen Geräten verbinden. Diese Bindung erfolgt bidirektional, indem Änderungen in dem Gerät in ABox-Updates und umgekehrt ABox-Änderungen in Änderungen des externen Gerät transformiert werden. Diese Art der Bindung ist ein Quasi-Standard, der in vielen gängigen Hausautomatisierungssystemen wie beispielsweise OpenHAB (www.openhab.org) universAAL (vgl. www.universaal.org) oder openURC (vgl. www.openurc.org) zum Einsatz kommt.

Der Vorteil dieses Verfahrens ist, dass die Protokollebene vor dem jeweiligen Assistenzprozess versteckt wird (dies funktioniert nur auf einer Ontologie-basierten Ebene). Der Austausch von Geräten oder Änderungen auf der Austauschprotokollebene ist ohne Änderung des (auf der ontologischen Ebene arbeitenden) Assistenzprozesses möglich. Geräte damit einfach hinzugefügt werden, indem die bestehende Ontologie wiederverwendet wird, wenn keine neuen relevanten Features zum Einsatz kommen. Ansonsten muss die Ontologie lediglich um die neuen Konzepte erweitert werden.

Die Erhaltung der Laufzeitkonsistenz ist eine der Hauptaufgaben eines offenen Systems zur Orchestrierung externer Prozesse und Geräte. Im SHIP-TOOL wird diese Frage der Konsistenz auf mehreren Ebenen adressiert. Zum einen werden durch die interne Repräsentation der Umgebung als eine Ontologie mögliche Inkonsistenzen sofort durch das zugrunde liegende DL-Inferenzsystem (Pellet) erkannt. Eine durch eine Basisoperation eingeführte Inkonsistenz führt zu einem Fehlschlag in der Prozesssemantik und kann entsprechend behandelt werden. Der Ansatz erlaubt außerdem eine Flexibilität in der Fehlerbehandlung: Eigenschaften, die in der Ontologie definiert sind, werden strikt behandelt, d.h. das DL-Inferenzsystem wird ihre Einhaltung immer erzwingen. Definiert man beispielsweise ein Konzept für Türen als eine disjunkte Vereinigung von offenen und geschlossenen Türen,

```
Door := Open | Closed
```

so kann eine Tür in der Ontologie unmöglich zugleich offen und geschlossen sein. Es kann aber Situationen geben, in dem diese Eigenschaft erwünscht ist, zum Beispiel wenn man nicht aus den Sensordaten nicht ihren Zustand ablesen kann. Für solche Fälle bietet sich eine Definition der Tür als eine (nicht notwendigerweise disjunkte) Vereinigung von offenen und geschlossenen Türen an

```
Door := Open + Closed
```

und spezifiziert einen Monitor zur Erkennung dieser unerwünschter Türzustände, in denen die Tür in beiden Zuständen ist:

```
monitor ProblematicDoor(d) = F(d:Open and d:Closed)
```

Wenn dieser Monitor erfolgreich terminiert, dann ist die Tür `d` in einem ungewissen Zustand und ein Prozess kann angestoßen werden, der zur Klärung der Situation beiträgt. Details über diese flexible Integration von Prozessen und Daten und dem Laufzeit-Konsistenzmanagement sind in [AHS13] zu finden.

Updates

Wie bereits erwähnt, bilden Updates das zentrale Programmierkonstrukt; insofern ist die effiziente Durchführung solcher Updates eine unabdingliche Voraussetzung für den Einsatz des SHIP-Tool in der Praxis. Daher wurde im Rahmen dieses Arbeitspaketes auf Empfehlung des wissenschaftlichen Beirats ein effizienter Update-Mechanismus für ABoxen entwickelt [AH13].

Dafür wurden in einem ersten Schritt die möglichen Spezifikationen für ABoxen auf sogenannte *konstruktive* Ontologien eingeschränkt. Hintergrund ist die Beobachtung, dass nahezu alle Fakten einer ABox in den Fallstudien Instanzen abstrakter Datentypen widerspiegeln. Bei einer Initialisierung einer solchen Instanz in Standardprogrammiersprachen möchte man alle Unterkomponenten dieses Datentyps mit konkreten Werten initialisieren. Analog dazu fordert man bei konstruktiven ABoxen, dass jede Wertänderungen eines Individuums eines Konzeptes über die Änderung der Werte der Subkonzepte erfolgt, so dass zu jeder Zeit die Komponenten des simulierten äbstrakten Datentypsëindeutig bestimmt sind. Ist beispielsweise ein Individuum für das Konzept `IDObject <ex at . AbstPosition` gegeben, so weiß man, dass *d* eine Position besitzt. Man fordert daher, dass damit die ABox auch eine Spezifikation der konkreten Position von *d* beinhaltet. Weiterhin fordert man für Disjunktionen von Konzepten $D \sqsubseteq E \sqcup F$, dass für jedes Individuum in *D* bekannt ist, ob es zu E oder F oder beiden gehört. Dies bedeutet, dass für jedes Individuum *d* eines (komplexen) Konzepts die ABox die vollständige Information über die Zusammensetzung und Werte der Attribute besitzt. Mit anderen Worten, kennt die ABox stets die Individuen, die für die Benennung der Werte der verschiedenen Attribute nötig sind,

womit die Erzeugung von Skolemfunktionen zur Benennung eines Attributwerts vermieden wird. Die gleiche Stringenz wird bei der Spezifikation der Rollen gefordert. SHIP erlaubt die Definition zusammengesetzter Rollen, z.B. $\mathrm{r} = \mathrm{r}_1 \cdot \mathrm{r}_2$. Weiß man jetzt, dass zwei Individuen a, b in der Relation r stehen, so muss es ein Individuum c geben, für das $(\mathrm{a}, \mathrm{c}) : \mathrm{r}_1$ und $(\mathrm{c}, \mathrm{b}) : \mathrm{r}_2$ gilt. Man fordert nun, dass das Individuum c explizit bekannt ist und die entsprechenden Zusammenhänge zwischen a, b, c in der ABox ableitbar sind.

Fordert man nun, dass ein System (TBox + RBox) und damit auch seine Zustände (ABox) mit Hilfe konstruktiver Ontologien spezifiziert sind, so sind alle Individuen quasi von unten nach oben (d.h. konstruktiv) definiert. Hat man ein Individuum a des oben genannten Konzepts D, so fordert die Konstruktivität zusätzliche Information über a. In vielen Fällen reicht diese ßusätzlicheÏnformation, z.B. $a : E$ aus, um $a : D$ abzuleiten, womit diese Ausgangsinformation redundant wird. Insbesondere besteht die Gefahr, dass mit Angabe, ob a dem Konzept D angehört, die Datenbasis potentiell inkonsistent werden kann (z.B. durch $\mathrm{a} : \overline{\mathrm{D}}$). Analog zu [Neb90] unterscheiden wir daher zwischen primitiven und definierten Konzepten und Rollen. Wir modifizieren eine gegebene TBox, indem wir die Gleichheit zwischen Konzepten explizit machen. Wann immer die TBox eine Menge von Axiomen der Form $c \sqsubseteq \mathrm{D}_i, i = 1, \ldots, n$ enthält, wird überprüft, ob ebenfalls $O \vdash \mathrm{D}_1 \ \& \ \ldots \ \& \ \mathrm{D}_n \sqsubseteq c$ gilt. In diesem Fall wird die Menge der Axiome durch die Gleichung $c = \mathrm{D}_1 \ \& \ \ldots \ \& \ \mathrm{D}_n$ ersetzt, sofern dies nicht gegen die Zyklenbedingung von TBox+RBox verstößt. Offensichtlich verändert diese Modifikation der TBox nicht die (semantischen) Modelle der Ontologie.

Auf Basis dieser normalisierten TBox wird die azyklische Abhängigkeitsrelation $>_c$ auf den Konzepten dazu verwendet, um die Definiertheit von Konzepten auszudrücken: Ein Konzept $c \in \Sigma_c$ ist definiert, wenn es ein Konzept $d \in \Sigma_c$ gibt, so dass $c >_c d$ gilt. Im anderen Fall ist c primitiv. Analog spezifiziert man definierte und primitive Rollen mit Hilfe der azyklischen Relation $>_r$ auf Rollen.

Für eine gegebene Ontologie O kann man nun die Redundanzfreiheit erzwingen, indem man fordert, dass alle ABox-Fakten über primitive Konzepte und Rollen gehen. Für eine Konstruktivität benötigt man, dass für alle Fakten über existenzquantifizierte Rollen es entsprechende deklarierte Individuen gibt, die diese Rollenbeziehung erfüllen.

Als Beispiel sei eine Ontologie für `List` in der SHIP-Syntax wie folgt gegeben:

```
List ::= EmptyList | NonEmptyList(hd:Elem,tl:List)
```

Internally, this is expanded to standard DL assertions, i.e.

```
List = EmptyList + NonEmptyList
Disjoint(EmptyList, NonEmptyList)
NonEmptyList < (ex hd . Elem) & (ex tl . List)
hd:NonEmptyList * Elem
Fun(hd)
tl:NonEmptyList * List
Fun(tl)
```

Das einzige primitive Konzept ist `EmptyList`. Primitive Rollen sind `hd` und `tl`. Die ABox beinhaltet als einziges das primitive Fakt `nil:EmptyList, (a,nil):tl` und ist damit nicht redundant aber nicht konstruktiv. Erst durch das Hinzufügen von `(a,e):hd` wird sie konstruktiv.

Die Updates einer ABox entsprechen den Zustandsübergängen des spezfizierten Systems. Diese Updates werden entweder durch Prozesse, die in dem System laufen, oder durch die Umgebung, die mit dem System interagiert, veranlasst. Unabhängig von dem Anlass des Updates fordert man, dass Updates stets die Konstruktivität der Ontologie bewahren. Hierfür müssen die Updates in nicht-redundanter Form ausschließlich mit Hilfe von ABox-Fakten über primitive Konzepte und Rollen spezifiziert sein. Die damit erhaltene Ontologie $\langle T, R, A' \rangle$ kann dennoch inkonsistent werden, wenn in der TBox spezifizierten Restriktionen über die Anzahl von Individuen eines Konzepts verletzt werden. In diesem Fall wird das Update nicht ausgeführt und die Ontologie bleibt unverändert. Wurde das Update durch eine Aktion veranlasst, schlägt die Aktion fehl. Wurde das Update durch die Umwelt veranlasst, so muss ein geeigneter Reparaturprozess spezifiziert sei, der dafür sorgt, dass das SHIP-TOOL und die Umwelt wieder synchronisiert werden. Ist die Ontologie konsistent, so ist sie nicht notwendigerweise konstruktiv. Daher wird die oben bereits erwähnte Konstruktion verwendet, um zu überprüfen, ob die erhaltene Ontologie $\langle T, R, A' \rangle$ konstruktiv ist. Falls nicht, stellt das SHIP-TOOL detaillierte Informationen zur Verfügung, welche Informationen fehlen. Dies ist insbesondere zu Testzwecken hilfreich, um die Effekte von Aktionen auf die definierten Prozesse hinsichtlich der Konstruktivität und Primitivität zu analysieren.

In manchen Fällen verursachen Updates Änderungen des Zustands, die nicht explizit durch die Spezifikation der Updates gegeben sind. In solchen Fällen müssen solche indirekten Konsequenzen eines Updates der Ontologie zusätzlich hinzugefügt werden, um reale und virtuelle Welt synchron zu halten. Dies muss auch spontan erfolgen, um künstliche Zwischenzeitpunkte, die durch ein eingeschobenes Update erzeugt würden, zu vermeiden. Das Problem der indirekten Konsequenzen ist in der Literatur als *"ramification problem"* bekannt. In SHIP wurde

der Ansatz der *"causal relationships"* [BLL10] gewählt, um diese indirekten Konsequenzen zu spezifizieren und die Ontologie gegebenenfalls um diese zu ergänzen.

2.1.5 Arbeitspaket AAL: Fallstudien

Auf Anraten des wissenschaftlichen Beirates (SAB) entwickelten wir das SmartTies-System [ADH+12], welches Standard- bzw. Norm-gerechte Entwicklungsprozesse für sicherheitskritische Software unterstützt, beispielsweise Common Criteria, DIN 26262 oder IEC 61508. Das SmartTies-System verwaltet alle in der Entwicklung anfallenden Dokumente und inklusive aller Abhängigkeiten. Es unterstützt die Zertifizierung des Prozesses indem es den Status von Dokumenten verwaltet, ob dieses geändert wurde, zur Begutachtung vorgelegt wurde, die externe Begutachtung erfolgreich war oder weitere Änderungen angemahnt wurden. SmartTies basiert auf dem DocTIP-System, das im Projekt FormalSafe entwickelt wurde, und erweitert dieses um spezifische Dokumenttypen, Regelmengen zur Analyse von Auswirkungen von Änderungen, spezielle Unterstützung des Entwicklungs- und Zertifizierungsprozesses sowie eine Web-basierte Schnittstelle (siehe auch Abbildung 2.4). Die meistens Dokumente wie die Konzeptpapier, die Fehleranalyse oder die Anforderungsbeschreibungen werden in MS Word geschrieben. SmartTies verwaltet die Dokumente intern in einer XML Dokumente, die diese explizit unterteilt, etwa Tabellen von Fehlerfällen und Sicherheitsanforderungen, deren die Beziehungen untereinander und zur Implementierung, Testfällen und Beschreibungen der Einsatzumgebungen. Dokumententyp-spezifische Parser transformieren die MS Word Dokument in die entsprechenden XML Strukturen, die SmartTies verwaltet und aus denen SmartTies wieder neue MS Word Dokumente erzeugen kann. Der Entwicklungsprozess wird unterstützt indem Dokumente verschiedene Stati durchlaufen, von *in Bearbeitung* während der Entwicklung bis *genehmigt* nach erfolgreicher Abnahme durch einen externen Gutachter. SmartTies verfolgt die Dokumentenstati, gewährleistet das jegliche Änderungen sorgfältig dokumentiert werden, und kümmert sich um die Versionskontrolle. Der Begutachtungsprozess ist ergänzt um ein einfaches "Ticketing"-System, womit Gutachter (interne und externe) ihre offenen Fragen an die Systementwickler einstellen können. Das SmartTies-System wurde auf der Cebit 2012 präsentiert.

Die in SmartTies realisierte Unterstützung eines Entwicklungsprozesses bietet auch eine zusätzliche Fallstudie für SHIP. Der Norm-gerechte Software-Entwicklungsprozess, zum Beispiel nach IEC 61508, ist a priori sehr rigide, da er fordert, dass die verschiedenen Phasen Fehleranalyse, Anforderungsbeschreibung, Spezifikation, Implementierung und Testen in strikt in

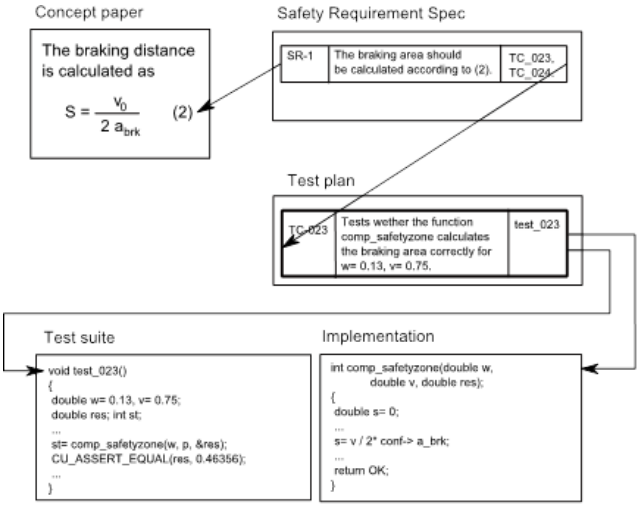

Abbildung 2.4 Beziehungen zwischen den mit SmartTies verwalteten Dokumenten

dieser Reihenfolge durchlaufen werden müssen. In der Praxis ist aber mehr Flexibilität im Prozessablauf notwendig, um Anforderungen anpassen zu können nachdem fehlende Fehlerfälle beim Testen aufgedeckt wurden. Um der Norm zu genügen müssen lediglich alle Änderungen nachverfolgbar sein. Für SHIP lieferte dies eine interessante Fallstudie sowohl den strikten Softwareentwicklungsprozess als auch den flexiblen Softwareentwicklungsprozess samt Änderungsinformationen zu formalisieren und zu beweisen, das flexible eine Verfeinerung des strikten ist in dem Sinn, dass jeder abgeschlossene Entwicklung im flexiblen Modell auch Ergebnis einer strikten Entwicklung wäre. Die Implementierung eines Unterstützungssystems des flexiblen Entwicklungsprozesses, wie das SmartTies-System, würde diese Eigenschaft erben.

Ein weiterer für SHIP wichtiger Aspekt ist, dass das Unterstützungssystem im Wesentlichen ein Monitoringsystem für den Entwicklungsprozess ist, ergänzt um Prozess zur Verwaltung der Dokumentenstati und Änderungsbeziehungen. Daraus folgte, dass der in SHIP zu entwickelnde Spezifikationsformalismus es erlauben muss sowohl ausführbare Prozesse als auch Monitoring Prozesse zu spezifizieren und zusätzliche Prozesse die ausgeführt werden, wenn der Gesamtprozess vom erwarteten Prozess abweicht, um diesen wieder normkonform zu machen.

Dies ist eine allgemeine Eigenschaft aller in SHIP betrachteten Prozesse, da diese immer mit Benutzern und Systemen (zum Beispiel die autonomen Rollstühle) interagieren, von denen ein "korrektes" Verhalten zwar erwartet aber nicht garantiert werden kann. Deswegen müssen

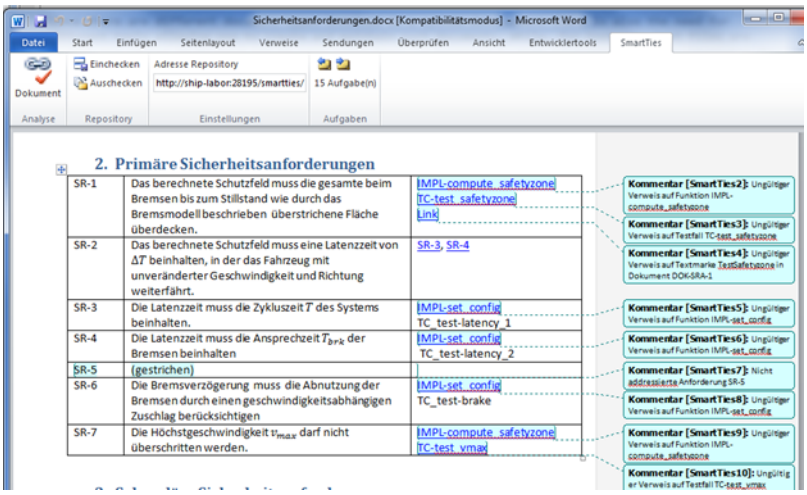

Abbildung 2.5 SmartTies-Integration in Microsoft Word

diese beobachtet werden und Aktionen beschrieben werden, wenn diese vom erwartenden Verhalten abweichen. Um Eigenschaften über das Gesamtsystem formulieren und beweisen zu können, müssen die Prozesse, die Monitore für erwartete Verhalten und die Reaktionen auf unerwartetes Verhalten in einem einheitlichen Formalismus beschrieben werden. Diese wurde im Arbeitspaket FF (Abschnitt 2.1.1) berücksichtigt. Die SHIP-Simulations-Plattform (Aufgabe VC[3], Abschnitt 2.1.3) erlaubt es Prozesse zusammen mit Monitoren in Interaktion mit Modellen der realen Umgebung zu simulieren.

Parallel wurde zu Beginn des Projektes mit der Spezifikation der Anwendungsszenarien für das Bremer Ambient Assisted Living Lab (BAALL) begonnen, die als durchgängige Testfälle für die zu entwickelnden Softwarepakete dienen sollten. Dabei wurden zwei Szenarien zur Evaluation der in SHIP entwickelten Formalismen, Methoden und Werkzeuge ausgearbeitet. Ziel war, dass dabei existierende Komponenten des BAALL und Daten aus unterschiedlichen Quellen und Formaten verwenden, und die entsprechende Assistenz erfordert eine Flexibilität der Prozesse, die über das hinausgeht, was Stand der Technik in AAL Umgebungen ist.

Das erste Szenario ist ein "Transport Szenario". Die Bewohner des BAALL die in ihrer Beweglichkeit eingeschränkt sind erhalten eine Transportassistenz mit Hilfe der autonom fahrenden Rollstühle, die eigenständig im BAALL navigieren können. Da sie aber voneinander keine Kenntnis haben und wir sie als unveränderliche "Black-Box"-Systeme ansehen, müssen sie durch einen Transportmanager koordiniert werden, damit Sie sich nicht gegenseitig so behindern, das keiner der Beiden mehr weiterfahren kann. Das zweite Szenario beschäftigt sich

Abbildung 2.6 Rolland auf dem Weg, einen Bewohner abzuholen

mit Kochassistenz, wobei das BAALL die gesamte Planung des Essens und die fristgerechte Zubereitung assistiert. Konkret werden Menüvorschläge unterbreitet, die kompatibel sind mit den gesundheitlichen Einschränkungen der Bewohner des BAALL und gleichzeitig die verfügbaren Lebensmittel am besten Ausschöpfen (zum Beispiel, dass Lebensmittel vor dem Mindesthaltbarkeitsdatum aufgebraucht werden oder das man so selten wie möglich Einkaufen gehen muss). Im Anschluss an die Menüauswahl wird die fristgerechte Zubereitung der einzelnen Speisen assistiert und begleitet, wobei auch parallel verschiedene Speisen zubereitet werden können sollen. Aus beiden Szenarien können jeweils eine reihe von konkretisierten Szenarien entwickelt werden, so dass man mit einem einfache Szenario anfangen kann und dieses sukzessive komplexer gestaltet werden können, zusammen mit deren Spezifikation und den Verifikationsaufgaben.

Zur Analyse des Stand der Technik wurden Softwareplattformen evaluiert, mit denen verteilte, kommunizierende Prozesse implementiert werden können. Beispiele hierfür sind JBoss[3] womit Business Prozesse modelliert werden können oder Jade[4] und Jason[5] zur Modellierung von Multiagentensystemen. Damit wurde ein Überblick gewonnen über bereits existierende

[3] http://www.jboss.org/
[4] http://jade.tilab.com/
[5] http://jason.sourceforge.net/Jason/Jason.html

Standards und Formalismen zur Beschreibung von heterogenen Systemen auf oder nahe der Implementierungsebene, insbesondere in die Eigenschaften und Konzepte der entsprechenden Plattformen und ob bzw. wie diese als Teil der BAALL-Plattform verwendet werden könnten. Ergebnis war, dass in SHIP eine höhere Abstraktionsebene als die AgentSpeak Sprache angestrebt werden sollte, die möglichst Nahe an die implementierungsnahen, untersten Spezifikationsebene aus Arbeitspaket FF heranreicht.

Prozesse und Monitore aus dem BAALL Szenarien und aus dem SmartTies-Entwicklungsprozess wurden im Spezifikationsformalismus des Arbeitspakets FF spezifiziert und in der SHIP Simulationsplattform, dem SHIP-TOOL, simuliert. Das SHIP-TOOL wurde anschließend an das reale BAALL und die autonomen Rollstühle Rolland angeschlossen. Hierfür wurden entsprechende Datenkonnektoren implementiert, was insbesondere für die Rollstühle etwas aufwändiger war. Hier bestand die Hauptarbeit darin ein geeignetes Synchronisationsmodell zwischen den realen Zuständen der Rollstühle und den Zuständen im ABox-Modell zu finden, welches berücksichtigt, dass die Verbindung zu den via WLAN angebundenen Rollstühlen zwischendurch abbrechen kann. Dabei zeigte sich in den realen Tests, dass es eine richtige Entscheidung war die Problematik abbrechender Funkverbindungen vollständig in dem Datenkonnektor zwischen Rollstuhl und Ontologie von den zentralen, koordinierenden Prozesse zu trennen. Das entworfene Transportszenario wurde vollständig realisiert mit zwei autonom fahrenden Rollstühlen in der Laborwohnung BAALL. Dieselben Prozessbeschreibungen können sowohl zur Simulation als auch zur Steuerung der realen BAALL Wohnung und Rollstühle verwendet werden, wobei lediglich die Datenkonnektor-Komponente ausgetauscht werden muss. Dies wurde im Frühjahr 2013 auf der Cebit 2013 in Hannover vorgeführt, wo das Notfallszenario und das Transportszenario vorgeführt wurden, die im SHIP-TOOL in Hannover ausgeführt wurden und dabei die BAALL Wohnung samt Rollstühlen in Bremen fernsteuerten. Dies konnten von den Besuchern auf der Cebit via Internet-Kameras verfolgt werden. Das Transportszenario wurde anschließend noch erweitert um Komfortassistenz, so dass die Türen entlang der Rollstuhlfahrten geöffnet und nach Durchfahrt wieder geschlossen werden und die durchfahrenen Bereiche der Wohnung, falls notwendig, beleuchtet werden. Diese Assistenz wird seitdem den regelmäßigen Besuchergruppen des BAALL vorgeführt.

2.1.6 Arbeitspaket S4A: SHIP4AAL: Integration mit einer Robotikumgebung

Im Oktober 2013 wurde eine weitere, komplexere Fallstudie begonnen, welche das im CAPIO Projekt entwickelte Exoskelett in die SHIP Assistenzprozesse integrierte. Hierfür wurde der

Abbildung 2.7 Fernsteuerung des humanoiden Roboter AILA durch das CAPIO-Exoskeleton

humanoide Roboter AILA[6] (Siehe Abbildung 2.7) in die BAALL Wohnung gebracht und logisch in die Ontologie des SHIP-TOOLs integriert: Das Szenario der Fallstudie bestand darin, dass eine behinderte Person den AILA Roboter mittels des Exoskeletts fernsteuern kann. Die behinderte Person sitzt im Wohnzimmer und möchte einen Schal, der in einem Regal im Kleiderschrank liegt. Die im SHIP-TOOL realisierten Assistenzprozesse organisieren, dass ein autonomer Rollstuhl beauftragt wird den Rollstuhl den Schal abzuholen, der vom Roboter AILA aus dem Schrank geholt und auf den Rollstuhl gelegt wird. Das Regal in dem der Schal liegt wird AILA durch das SHIP-TOOL durch ein blinkendes Licht angezeigt. Die behinderte Person assistiert AILA mittels Exoskelett beim Greifen des Schals im Regal (siehe Abbildung 2.8). Dies geschieht mittels Videobild aus den Kamera-Augen von AILA auf den Fernseher. Der gesamte Aufbau und Ablauf des Assistenzszenarios wurde in einem Video dokumentiert. Für SHIP zeigte diese zusätzliche Fallstudie, dass die zu dem Zeitpunkt schon existierenden Prozesse und Ontologien der anderen Fallstudien einfach erweitert werden konnten: Die Ontologie wurde erweitert um eine Ontologie für Kleiderschrank, Regale, Regallichter und Kleider. Kleider sind Objekte, die einen Ort haben und durch Rollstühle transportiert werden können. Damit konnten alle Transportassistenzprozesse unverändert übernommen werden. Der Prozess zur Steuerung der Regallichter war eine einfacher, zusätzlicher, parallel laufender Prozess, der lediglich in der aktuellen Ontologie überprüft, ob ein angefordertes Kleidungsstück in einem Regal liegt und entsprechend die Beleuchtung an oder abschaltet.

Schließlich wurden das SHIP-TOOL auch außerhalb des Projekts in einem anderen Projekt verwenden, um die Einhaltung klinischer Leitlinien in Krankenhäusern zu überwachen und

[6] http://robotik.dfki-bremen.de/de/forschung/robotersysteme/aila.html

Abbildung 2.8 AILA greift einen Schal

Leitlinien-konforme Behandlungsvorschläge zu unterbreiten (siehe Abschnitt 2.4.3). Hierbei wurden die Vorgaben klinischer Leitlinien in Monitoren spezifiziert, die die verschiedenen Phasen einer medizinischen Behandlung beobachten. Weicht eine Behandlung hiervon ab werden entsprechende Prozesse initiiert, so dass die Behandlung insgesamt wieder Leitlinien-konform ist.

2.2 Wichtigste Positionen des zahlenmäßigen Nachweises

Siehe Anlage.

2.3 Notwendigkeit und Angemessenheit der geleisteten Arbeit

Die Arbeiten zum Erreichen des Vorhabens wurden in mehreren ihrerseits wieder strukturierten Arbeitspaketen detailliert geplant und bildeten die Grundlage für die angegebenen Meilensteine. Der Verlauf der Arbeit im Projekt erfolgte gemäß der im Projektantrag formulierten Planung. Es waren keine zusätzlichen Ressourcen für die geplanten Arbeiten des Projekts nötig.

2.4 Verwertbarkeit

2.4.1 Wirtschaftliche Erfolgsaussichten

Bereits während der Durchführung des Projektes zeigte sich, dass das anvisierte System ein großes ökonomisches Potential haben würde, da es auf eine Methodik sowie entsprechende Techniken und Werkzeuge abzielt, die eine semantik-basierte Integration heterogener Prozesse unterstützen. Dies wird insbesondere auch an den bereits anlaufenden industriellen Kooperationen deutlich, die im Umfeld eines möglichen Einsatzes des SHIP-TOOL entstanden sind.

Die mit dem Paradigma der serviceorientierten Architekturen begonnene Integration verschiedener Systeme hinsichtlich einer gemeinsamen Datensicht dehnt sich zunehmend auch auf die Komposition und Integration verschiedenartiger Arbeitsabläufe bzw. Prozesse aus. Die wechselseitigen Abhängigkeiten der ablaufenden Prozesse und die Komplexität der durch sie verarbeiteten Informationen erschweren zunehmend die bisher weit verbreitete manuelle Vermittlung zwischen verschiedenartigen Prozessen. Dies zeigt sich insbesondere auch in den aus der Industrie aufkommenden Fragestellungen. Ein Beispiel ist die Entwicklung eingebetteter Systeme, in denen mechanische, elektrotechnische und softwarespezifische Prozesse aufeinander abgestimmt entwickelt werden müssen.

Ein erste Anwendung des SHIP-TOOL ausserhalb des SHIP-Projektes erfolgte in dem mit der Firma ID GmbH in Berlin durchgeführten BMBF-Projekt SIMPLE. Das Kernziel von SIMPLE war eine experimentelle Implementierung von klinischen Leitlinien in enger Verbindung mit bestehenden Krankenhausinformationssystemen. Die entstandene Lösung einer Modellierung und Umsetzung der Leitlinien auf Basis der in SHIP entwickelten Techniken und Werkzeuge erwies sich als äußerst tragfähig hinsichtlich des Abdeckungsgrades der einzelnen Leitlinienaspekte. Bei den exemplarisch untersuchten Leitlinien für die Behandlung bei akuten Herzkrankheiten konnten nicht nur die üblichen Behandlungsabläufe (Workflows) semantisch fundiert repräsentiert werden, sondern auch andere dazu parallel stattfindende Aktivitäten, wie beispielsweise die Vergabe von Medikamenten und die Überwachung, ob eventuell unerwünschte Nebenwirkungen auftreten. Der in SIMPLE entwickelte Ansatz fand dabei in mehreren internationalen Kongressen großes Interesse sowohl bei akademischen Kollegen als auch bei Medizintechnikherstellern.

Die in SHIP entwickelten Technologien und insbesondere die Quellen von SHIP sind unter der GPL-Lizenz, die einen freien wissenschaftlichen Gebrauch erlaubt, veröffentlicht. Dies garantiert eine weite Sichtbarkeit und ermöglicht es insbesondere den mit uns kooperierenden Partnern, den Korpus an Formalismen und Werkzeugen zu erweitern und die Technologien in neuen Anwendungsbereichen einzusetzen.

2.4.2 Wissenschaftliche Erfolgsaussichten

Die wissenschaftlichen und technischen Erfolgsaussichten des Projektes SHIP werden als hoch eingestuft. Die beantragten Arbeiten beruhen auf Kernkompetenzen des DFKIs in den Gebieten Formale Methoden, Semantik und Dokumenten-Management. Sie basieren auf zahlreichen früheren Projekten, die von dritter Seite aber auch besonders durch das BMBF (s. Abs. 1.4.1) gefördert wurden, und garantieren daher jene Kontinuität in Forschung und Entwicklung, die hohe wissenschaftliche und technische Erfolgsaussichten gewährleistet.

SHIP basiert auf dem Wissen und der Expertise, die in den letzten zehn Jahren auf diesen Gebieten gesammelt wurden, und wird es dem DFKI erlauben, seine Kompetenz und seine Position in diesen Gebieten in der Zukunft zu stärken. Die semantische Integration heterogener Prozesse, die insbesondere auch komplexe Informationen verarbeiten, ist bereits jetzt ein hoch-innovatives und wirtschaftlich wichtiges Thema für das DFKI.

Das DFKI als Mitglied in verschiedenen Networks of Excellence und die starke Verankerung zahlreicher Projektmitglieder in entsprechenden wissenschaftlichen Gremien und Kreisen werden eine große Verbreitung und Sichtbarkeit der erzielten wissenschaftlichen und technologischen Ergebnisse im Sinne von Veröffentlichungen, Präsentationen auf Konferenzen, oder prototypischen Implementierungen garantieren.

Der Kontakt zu dem Stand der Wissenschaft auf diesem Gebiet und zu seiner Fortentwicklung wird durch die intensiven anwendungsorientierten Forschungstätigkeiten des DFKIs in diesen Bereichen gewährleistet. Dies betrifft sowohl den Anwendungsbereich AAL, in dem das DFKI insbesondere über eine entsprechende Musterwohnung (BAALL) verfügt, als auch in den Bereichen Formale Methoden und Prozessmodellierung.

2.4.3 Wissenschaftlich-wirtschaftliche Anschlussfähigkeit

SHIP basiert auf dem Wissen und der Expertise, das die Arbeitsgruppe im DFKI in den letzten Jahren angesammelt hat, und erlaubt es dem DFKI, seine Kompetenz und Stellung in diesen Bereichen zu festigen und zu verbessern. Die semantische Verbindung heterogener Prozesse inklusive der darin bearbeiteten Informationen bildet eine der zentralen Herausforderungen dieser Dekade. Dies zeigt sich insbesondere in der Vielfalt der zur Zeit verfolgten Anwendungsgebiete, die weit über die in diesem Projekt betrachtete AAL-Umgebung hinausgehen. Im Nachfolgenden skizzieren wir einige dieser Aktivitäten.

Klinische Leitlinien Im Rahmen eines zur Zeit beantragten Projektes möchte unser Projektpartner ID GmbH in Berlin mit uns den im vorangegangenen Abschnitt skizzierten Ansatz zur Unterstützung Klinischer Leitlinien zu einem kommerziellen Prototyp weiterentwickeln. Schwerpunkte der Weiterentwicklung sind dabei die Skalierbarkeit des dem Ansatz zugrunde liegenden SHIP-tool und damit der effizienten Verarbeitung größerer ontologischer Faktenbasen, einer problemadäquaten Benutzerschnittstelle, die den zeitkritischen Anforderungen des Einsatzgebietes Rechnung trägt, und der semi-automatisierten Transformation von (formalisierten) Leitlinienspezifikationen in entsprechende Monitor- und Prozessbeschreibungen des SHIP-tool.

Sichere Mensch-Maschine-Interaktion Im Rahmen des BMBF-Projekts *HYSOCIATEA* sollen Techniken entwickelt werden, die es erlauben, Roboter und virtuelle Agenten mit dem Menschen in einem Team zusammenzuarbeiten. Ziel ist eine weitgehende Flexibilisierung der Aufteilung der an das Team gestellten Aufgaben, um die Vorteile der jeweiligen Teammitglieder auszunützen. Zur Verwirklichung dieses Zieles wird ein bestimmtes Maß an kognitiven Fähigkeiten den Robotern abverlangt. Sie müssen eine Situation erkennen können, die Handlungen der anderen Beteiligten erkennen und in die Situation einordnen können. Die in SHIP entwickelten Techniken der Verwaltung eines logik-basierten Zustands und des Programmierparadigmas, in dem Phasen der aktiven Handlung sich mit Phasen des aktiven Beobachtens abwechseln, bilden eine ideale Basis für die Modellierung der gewünschten kognitiven Fähigkeiten der einzelnen Roboter.

Smart Home and Intelligent Environments. Während der Projektlaufzeit ist die quelloffene Heimautomatisierungsplattform OpenHAB entstanden sowie die mehr Geschäftsmodell-orientierte UniversAAL Platform (Siehe 2.5). Analog zum SHIP-tool versuchen sie die Steuerungs- und Kommunikationsebenen von spezifischen Protokollen zu trennen. Das SHIP-tool geht, als Forschungsprototyp, konzeptionell über diese System hinaus indem es eine mächtige und vergleichsweise einfache Sprache zur Beschreibung von Automatisierungsprozessen bereitstellt. Hier ist großes Potenzial diese Umgebungen zu komplettieren und gleichzeitig an einem plattformunabhängigen Standard zur Beschreibung von Assistenzprozessen mitzuwirken. Ausgangspunkt hierfür ist die durch die Arbeiten in SHIP motivierte Beteiligung im VDI Arbeitskreis "AAL-Interoperabilität", der sich mit der Entwicklung von Interoperabilitätsstandards für AAL-Systeme beschäftigt.

2.5 Bekannt gewordener Fortschritt

Beschreibungslogiken

Im Bereich der Beschreibungslogiken gab es während der Projektlaufzeit von SHIP eine Reihe von Entwicklungen, die sich speziell auf das Design und die Implementierung des SHIP-tool ausgewirkt haben:

In [BGL12] wird eine Spezifikationssprache vorgestellt, die es erlaubt mit Hilfe von temporal-logischen Operatoren über die zeitliche Gültigkeit von ABox-Fakten Aussagen zu treffen. Als Temporallogik wird LTL verwendet, die Beschreibungslogik ist ALC. Die dort vorgestellte Logik bildete den Ausgangspunkt für die Entwicklung der Monitore im SHIP-tool.

In [BF12] wird die Fortschreibung temporal-logischer Formeln über diskrete Zeitticks (dies entspricht den Zeitpunkten der Updates in dem SHIP-Szenario) beschrieben. Der Ansatz wurde im Bereich des Planens entwickelt, um Anforderungen an einen zu generierenden Plan stellen zu können. Die temporal-logischen Formeln spezifizieren dabei Anforderungen an die Sequenz der Zustände, die bei Planausführung durchlaufen werden müssen. Im SHIP-tool wird dieser Ansatz zur Fortschreibung der Monitore verwendet.

In [CSGZ12] wurden verschiedene Beschreibungslogiken um Aktionen und Programmkonstrukte (z.B. "Sequence", "Choice", "Any-Order", "Iterate", "If-Then-Else", "Repeat-While" und "Repeat-Until") erweitert, um letztlich eine auf eine Beschreibungslogik agierende Programmiersprache zu erhalten. Für die Verifikation dieser dynamischen Beschreibungslogiken (DLL) wurde ein spezieller Tableauxkalkül entwickelt.

Haus Automatisierungsplatformen

Im Bereich der Heimautomatisierungsplatformen gab es im Projektzeitraum bei industriellen Anbietern keinen nennenswerten Fortschritt. Relevant für die SHIP verfolgte Ziele sind die OpenHAB Platform (`http://www.openhab.org/`) und die UniversAAL Platform (`www.universaal.org`), auf die im Folgenden eingegangen wird.

OpenHAB ist eine quelloffene Platform zur Hausautomatisierung, die von spezifischen Steuerungs und Busprotokollen abstrahiert und eine vielzahl von Anschlüßen zu speziellen Systemen bereitstellt. Die Signale der Sensoren und Aktoren werden abstrahiert als Events einen Bus geschickt und analog werden Steuerungssignale auch als Events wieder rausgeschickt. Diese können innerhalb von OpenHAB konfiguriert werden und eine einfache, von Drools (`http://drools.jboss.org/`) inspierierte Regelsprache erlaubt die Definition von einfachen Automatisierungsregeln, wie auf Events reagiert werden kann. Im Gegensatz zum SHIP-tool gibt es keine konsistente, ontologische Beschreibung der Zustände und keine, höhere Programmierebene für Prozesse. Ein Monitoring von Verhalten der Umgebung wie im SHIP-tool fehlt.

UniversAAL ist eine in einem EU Projekt entwickelt Platform für den AAL Bereich, der auch von den spezifischen Protokollen abstrahiert und auf einer ontologischen Repräsentation der Objekte und Events aufbaut. Diese werden im wesentlichen zur Repräsentation der Daten in Events aus bzw. in die Sensoren und Aktoren verwendet und in einer RDF Datenbank gespeichert. Diese können dann mittels Datenbank-ähnlichen Anfragen durchsucht werden. Eine konsistente Repräsentation des aktuellen Zustands wie im SHIP-tool fehlt. Darüberhinaus bietet UniversAAL auch eine Drools basierte Möglichkeit Regeln zu definieren. Eine höhere Programmierebene fehlt ebenso wie in OpenHAB sowie die Möglichkeit des Monitorings wie im SHIP-tool, könnte aber ebenfalls ergänzt werden indem man das SHIP-tool daran anschließt. Ansonsten liegt der Schwerpunkt in UniversAAL auf einfacher Interoperabilität und vor allem auf einem damit verbunden Konzept AAL-Applikation zu entwickeln und zu vermarkten.

2.6 Veröffentlichungen

[ADD+11] Autexier, Serge ; David, Catalin ; Dietrich, Dominik ; Kohlhase, Michael ; Zholudev, Vyacheslav: Workflows for the Management of Change in Science, Technologies, Engineering and Mathematics. In: Davenport, James H. (Hrsg.) ; Farmer, William (Hrsg.) ; Rabe, Florian (Hrsg.) ; Urban, Joseph (Hrsg.): *Proceedings of Calculemus/MKM 2011*, Springer, LNAI 6824, 2011, S. 164 – 179

[ADH+12] Autexier, Serge ; Dietrich, Dominik ; Hutter, Dieter ; Lüth, Christoph ; Maeder, Christian: SmartTies - Management of Safety-Critical Developments. In: Margaria, Tiziana (Hrsg.) ; Steffen, Bernhard (Hrsg.): *Proceedings 5th International Symposium On Leveraging Applications of Formal Methods, Verification and Validation (ISoLa'12)*. Amirandes, Heraclion, Crete : Springer, october 2012 (LNCS)

[ADL12] Aspinall, David ; Denney, Ewen ; Lüth, Christoph: Querying Proofs. In: *18th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR-18)* Bd. 7180, Springer, 2012 (Lecture Notes in Computer Science), S. 92–106

[AH12] Autexier, Serge ; Hutter, Dieter: Structure Formation to Modularize Ontologies. In: Schneider, Thomas (Hrsg.) ; Walther, Dirk (Hrsg.): *6th International Workshop on Modular Ontologies* Bd. Vol-875. Graz, Austria, july 2012 (CEUR)

[AH13] Autexier, Serge ; Hutter, Dieter: Constructive DL update and reasoning for modeling and executing the orchestration of heterogenous processes. In: *Proceedings of the 26th International Workshop on Description Logics, 2013. International Workshop on Description Logics (DL-13), July 23-26, Ulm, Germany* Bd. 1014, CEUR Workshop Proceedings, http://ceur-ws.org/Vol-1014/, 7 2013, S. 501–512

[AHBW13] Autexier, Serge ; Hutter, Dieter ; Bawadekji, Mohamed ; Wolters, Regine: Supporting Clinical Guidelines Using DL-Temporal Reasoning. In: *Proceedings of the 10th International Conference & Expo on Emerging Technologies for a Smarter World (CEWIT-2013), 10th, October 21-22, Melville,, NY, USA* Stony Brook University NY, IEEE Xplore Online, 10 2013

[AHMS13] Autexier, Serge ; Hutter, Dieter ; Mandel, Christian ; Stahl, Christoph: SHIP-Tool

Live: Orchestrating the Activities in the Bremen Ambient Assisted Living Lab (Demo). In: Augusto, Juan C. (Hrsg.) ; Wichert, Reiner (Hrsg.): *Proceedings of the Fourth International Joint Conference on Ambient Intelligence. International Joint Conference on Ambient Intelligence (Aml-2013), December 3-5, Dublin, Ireland*, Springer-Verlag, CCIS, 12 2013

[AHS13] Autexier, Serge ; Hutter, Dieter ; Stahl, Christoph: An Implementation, Execution and Simulation Platform for Processes in Heterogeneous Smart Environments. In: Augusto, Juan C. (Hrsg.) ; Wichert, Reiner (Hrsg.): *Proceedings of the Fourth International Joint Conference on Ambient Intelligence. International Joint Conference on Ambient Intelligence (Aml-2013), December 3-5, Dublin, Ireland*, Springer-Verlag, CCIS, 12 2013

[AHed] Autexier, Serge ; Hutter, Dieter: Structure Formation in Large Theories. In: *Journal of Automated Reasoning* (submitted)

[CHJ+13] Codescu, Mihai ; Horozal, Fulya ; Jakubauskas, Aivaras ; Mossakowski, Till ; Rabe, Florian: Compiling Logics. In: Narciso Martí-Oliet, Miguel P. (Hrsg.): *Recent Trends in Algebraic Development Techniques, 21th International Workshop, WADT 2012* Bd. 7841, Springer, 2013 (LNCS), S. 111–126

[CKP+11] Cirstea, Corina ; Kurz, Alexander ; Pattinson, Dirk ; Schröder, Lutz ; Venema, Yde: Modal logics are coalgebraic. In: *The Computer Journal* 54 (2011), S. 31 – 41

[CLMMar] Codescu, Mihai ; Langenstein, Bruno ; Maeder, Christian ; Mossakowski, Till: The VSE Refinement Method in Hets. In: *Electronic Communications of the EASST* (to appear)

[Die11] Dietrich, Dominik: *Proof Planning with Compiled Strategies*, FR 6.2 Informatik, Saarland University, Doctoral Thesis, 2011

[GBJLS11a] Gutierrez-Basulto, Victor ; Jung, Jean C. ; Lutz, Carsten ; Schröder, Lutz: A Closer Look at the Probabilistic Description Logic Prob-EL. In: Burgard, Wolfram (Hrsg.) ; Roth, Dan (Hrsg.): *Proceedings of the 25th Conference on Artificial Intelligence (AAAI-11)*, AAAI Press, Menlo Park, CA, 2011

[GBJLS11b] Gutierrez-Basulto, Victor ; Jung, Jean C. ; Lutz, Carsten ; Schröder, Lutz: The

Complexity of Probabilistic EL. In: Rudolph, Sebastian (Hrsg.) ; Zakharyaschev, Michael (Hrsg.): *Proceedings of the 24th International Workshop on Description Logics (DL 2011)*, CEUR-WS online proceedings, 2011

[GS11a] Goncharov, Sergey ; Schröder, Lutz: A coinductive calculus for asynchronous side-effecting processes. In: Owe, Olaf (Hrsg.) ; Steffen, Martin (Hrsg.) ; Telle, Jan A. (Hrsg.): *Fundamentals of Computation Theory (FCT 2011)* Bd. 6914, Springer, 2011 (Lecture Notes in Computer Science), 276-287

[GS11b] Goncharov, Sergey ; Schröder, Lutz: A Counterexample to Tensorability of Effects. In: Corradini, Andrea (Hrsg.) ; Klin, Bartek (Hrsg.): *Algebra and Coalgebra in Computer Science (CALCO 2011)*, Springer, 2011 (Lecture Notes in Computer Science), S. 208–221

[JKMR13] James, Phillip ; Knapp, Alexander ; Mossakowski, Till ; Roggenbach, Markus: Designing Domain Specific Languages - A Craftsman's Approach for the Railway Domain using CASL. In: Narciso Martí-Oliet, Miguel P. (Hrsg.): *Recent Trends in Algebraic Development Techniques, 21th International Workshop, WADT 2012* Bd. 7841, Springer, 2013 (LNCS), S. 178–194

[LKMG12] Lange, Christoph ; Kutz, Oliver ; Mossakowski, Till ; Grüninger, Michael: The Distributed Ontology Language (DOL): Ontology Integration and Interoperability Applied to Mathematical Formalization. In: Jeuring, Johan (Hrsg.) ; Campbell, John A. (Hrsg.) ; Carette, Jacques (Hrsg.) ; Reis, Gabriel D. (Hrsg.) ; Sojka, Petr (Hrsg.) ; Wenzel, Makarius (Hrsg.) ; Sorge, Volker (Hrsg.): *Intelligent Computer Mathematics - 11th International Conference, AISC 2012, 19th Symposium, Calculemus 2012, 5th International Workshop, DML 2012, 11th International Conference, MKM 2012, Systems and Projects, Held as Part of CICM 2012, Bremen, Germany, July 8-13, 2012. Proceedings* Bd. 7362, Springer, 2012 (Lecture Notes in Computer Science). – ISBN 978–3–642–31373–8, S. 463–467

[OMR10] O'Reilly, Liam ; Mossakowski, Till ; Roggenbach, Markus: Compositional Modelling and Reasoning in an Institution for Processes and Data. In: Mossakowski, Till (Hrsg.) ; Kreowski, Hans-Jörg (Hrsg.): *Recent Trends in Algebraic Development Techniques - 20th International Workshop, WADT 2010, Etelsen, Germany, July 1-4, 2010, Revised Selected Papers* Bd. 7137, Springer, 2010 (Lecture Notes in Computer Science). – ISBN 978–3–642–28411–3, S. 251–269

[SP11a] Schröder, Lutz ; Pattinson, Dirk: Description Logics and Fuzzy Probability. In: Walsh, Toby (Hrsg.): *International Joint Conference on Artificial Intelligence (IJCAI 2011)*, AAAI Press, Menlo Park, CA, 2011. – accepted for oral presentation

[SP11b] Schröder, Lutz ; Pattinson, Dirk: Modular Algorithms for Heterogeneous Modal Logics via Multi-Sorted Coalgebra. In: *Math. Struct. Comput. Sci.* 21 (2011), Nr. 2, S. 235 – 266

Literaturverzeichnis

[AH05] Autexier, Serge ; Hutter, Dieter: Mind the Gap - Maintaining Formal Developments in MAYA. In: *Festschrift in Honor of J.H. Siekmann*. Springer-Verlag, LNCS 2605, 2005

[AHK02] Alur, Rajeev ; Henzinger, Thomas A. ; Kupferman, Orna: Alternating-time temporal logic. In: *J. ACM* 49 (2002), 672–713. http://doi.acm.org/10.1145/585265.585270

[AHMS02] Autexier, Serge ; Hutter, Dieter ; Mossakowski, Till ; Schairer, Axel: The Development Graph Manager MAYA (system description). In: Kirchner, Helene (Hrsg.): *Proceedings of 9th International Conference on Algebraic Methodology And Software Technology (AMAST'02)*, Springer Verlag, 2002 (LNCS 2422)

[AM10] Autexier, Serge ; Müller, Normen: *Semantics-Based Change Impact Analysis for Heterogeneous Collections of Documents*. submitted, 2010

[BCM+03a] Baader, Franz (Hrsg.) ; Calvanese, Diego (Hrsg.) ; McGuinness, Deborah L. (Hrsg.) ; Nardi, Daniele (Hrsg.) ; Patel-Schneider, Peter F. (Hrsg.): *The Description Logic Handbook*. Cambridge University Press, 2003. – ISBN 0–521–78176–0

[BCM+03b] Baader, Franz (Hrsg.) ; Calvanese, Diego (Hrsg.) ; McGuinness, Deborah L. (Hrsg.) ; Nardi, Daniele (Hrsg.) ; Patel-Schneider, Peter F. (Hrsg.): *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003 . – ISBN 0–521–78176–0

[BF12] Bauer, Andreas K. ; Falcone, Yliès: Decentralised LTL Monitoring. In: Giannakopoulou, Dimitra (Hrsg.) ; Méry, Dominique (Hrsg.): *FM 2012: Formal Methods - 18th International Symposium, Paris, France, August 27-31, 2012. Proceedings*

Bd. 7436, Springer, 2012 (Lecture Notes in Computer Science). – ISBN 978–3–642–32758–2, S. 85–100

[BGL12] Baader, Franz ; Ghilardi, Silvio ; Lutz, Carsten: LTL over description logic axioms. In: *ACM Transactions on Computational Logic* (2012). http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:LTL+over+Description+Logic+Axioms#0

[BLL10] Baader, Franz ; Lippmann, Marcel ; Liu, Hongkai: Using Causal Relationships to Deal with the Ramification Problem in Action Formalisms Based on Description Logics. In: Fermüller, Christian G. (Hrsg.) ; Voronkov, Andrei (Hrsg.): *Logic for Programming, Artificial Intelligence, and Reasoning - 17th International Conference, LPAR-17, Yogyakarta, Indonesia, October 10-15, 2010. Proceedings* Bd. 6397, Springer, 2010 (Lecture Notes in Computer Science). – ISBN 978–3–642–16241–1, S. 82–96

[BM04] Bidoit, Michel ; Mosses, Peter D.: *LNCS (IFIP Series)*. Bd. 2900: Casl *User Manual*. Springer, 2004. http://dx.doi.org/10.1007/b11968. http://dx.doi.org/10.1007/b11968

[BP08] Benzmüller, Christoph ; Paulson, Lawrence: Exploring Properties of Normal Multimodal Logics in Simple Type Theory with LEO-II. In: Benzmüller, Christoph (Hrsg.) ; Brown, Chad E. (Hrsg.) ; Siekmann, Jörg (Hrsg.) ; Statman, Richard (Hrsg.): *Festschrift in Honor of Peter B. Andrews on His 70th Birthday*. College Publications, 2008 (Studies in Logic, Mathematical Logic and Foundations)

[BTPF08] Benzmüller, Christoph ; Theiss, Frank ; Paulson, Larry ; Fietzke, Arnaud: LEO-II - A Cooperative Automatic Theorem Prover for Higher-Order Logic. In: Armando, Alessandro (Hrsg.) ; Baumgartner, Peter (Hrsg.) ; Dowek, Gilles (Hrsg.): *Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings* Bd. 5195, Springer, 2008 (LNCS), 162-170

[CKP09] Cîrstea, Corina ; Kupke, Clemens ; Pattinson, Dirk: EXPTIME Tableaux for the Coalgebraic μ-Calculus. In: *Computer Science Logic, CSL 09* Bd. 5771, Springer, 2009 (LNCS). – ISBN 978–3–642–04026–9, 179–193

[CLMM09] Codescu, Mihai ; Langenstein, Bruno ; Maeder, Christian ; Mossakowski, Till: The

VSE Refinement Method in HETS. In: Breitman, K. (Hrsg.) ; Cavalcanti, A. (Hrsg.): *ICFEM 2009* Bd. 5885, Springer, 2009 (Lecture Notes in Computer Science), 660-678

[CM08] Codescu, Mihai ; Mossakowski, Till: Heterogeneous colimits. In: Boulanger, Frédéric (Hrsg.) ; Gaston, Christophe (Hrsg.) ; Schobbens, Pierre-Yves (Hrsg.): *MoVaH'08 Workshop on Modeling, Validation and Heterogeneity*, IEEE press, 2008

[CMPS09] Calin, Georgel ; Myers, Rob ; Pattinson, Dirk ; Schröder, Lutz: CoLoSS: The Coalgebraic Logic Satisfiability Solver (System Description). In: Areces, Carlos (Hrsg.) ; Demri, Stephane (Hrsg.): *Methods for Modalities (M4M-5, 2007)* Bd. 231, Elsevier, 2009 (Electron. Notes Theoret. Comput. Sci.), S. 41–54

[CoF04] CoFI (The Common Framework Initiative): *LNCS (IFIP Series)*. Bd. 2960: Casl *Reference Manual*. Springer, 2004. http://dx.doi.org/10.1007/b96103. http://dx.doi.org/10.1007/b96103

[CSGZ12] Chang, Liang ; Shi, Zhongzhi ; Gu, Tianlong ; Zhao, Lingzhong: A Family of Dynamic Description Logics for Representing and Reasoning About Actions. In: *J. Autom. Reasoning* 49 (2012), Nr. 1, S. 1–52

[DFHJ05] Denny, Ewen ; Fischer, Bernd ; Hutter, Dieter ; Jones, Mark: *Proceedings of the 2005 Automated Software Engineering Workshop on Software Certificate Management (SoftCeMent05)*. Long Beach, USA, November 2005

[FKS10] Franke, Marco ; Klein, Patrick ; Schröder, Lutz: Ontological Semantics of Standards and PLM Repositories in the Product Development Phase. In: *Proc. 20th CIRP Design Conference 2010*, 2010. – To appear

[GB92] Goguen, J. A. ; Burstall, R. M.: Institutions: Abstract Model Theory for Specification and Programming. In: *Journal of the Association for Computing Machinery* 39 (1992), S. 95–146. – Predecessor in: LNCS 164, 221–256, 1984.

[GBG+06] Geiss, Rubino ; Batz, Gernot V. ; Grund, Daniel ; Hack, Sebastian ; Szalkowski, Adam: GrGen: A fast SPO-based graph rewriting tool. In: *Graph Transformations - ICGT 2006. Lecture Notes In Computer Science*, Springer, 2006, S. 383–397

[GKPS10] Gore, Rajeev ; Kupke, Clemens ; Pattinson, Dirk ; Schröder, Lutz: Global Caching for Coalgebraic Description Logics. In: Giesl, Jürgen (Hrsg.) ; Haehnle, Reiner (Hrsg.): *International Joint Conference on Automated Reasoning, IJCAR 2010*, Springer, 2010 (LNCS). – To appear

[Het] *Hets: The Heterogeneous Tool Set*. Web site at `http://www.informatik.uni-bremen.de/agbkb/forschung/formal_methods/CoFI/hets/`,

[HS01] Hutter, Dieter ; Schairer, Axel: Towards an Evolutionary Formal Software Development. In: *Proceedings 16th IEEE International Conference on Automated Software Engineering, ASE-2001*. San Diego, USA : IEEE Computer Society, 2001

[HSar] Hausmann, Daniel ; Schröder, Lutz: Optimizing Conditional Logic Reasoning within CoLoSS. In: Bolander, Thomas (Hrsg.) ; Braüner, Torben (Hrsg.): *Methods for Modalities (M4M-6, 2009)*, Elsevier, To appear (Electron. Notes Theoret. Comput. Sci.)

[Hut00] Hutter, Dieter: Management of Change in Structured Verification. In: *Proceedings 15th IEEE International Conference on Automated Software Engineering*, IEEE Computer Society, 2000 (ASE 2000), S. 23–34

[Hut04] Hutter, Dieter: Towards a Generic Management of Change. In: *Workshop on Computer-Supported Mathematical Theory Development, International Joint Conference on Automated Reasoning´04*. Cork, Ireland, 2004

[Hut09] Hutter, Dieter: Semantic Management of Heterogeneous Documents. In: *Proceedings of the Mexican International Conference on Artificial Intelligence (MICAI-2009)*, Springer, LNAI 5845, 2009

[KBLL+04] Krieg-Brückner, B. ; Lindow, A. ; Lüth, C. ; Mahnke, A. ; Russell, G.: Semantic Interrelation of Documents via an Ontology. In: Engels, G. (Hrsg.) ; Seehusen, S. (Hrsg.): *DeLFI 2004, Tagungsband der 2. e-Learning Fachtagung Informatik, 6.-8. September 2004, Paderborn, Germany* Bd. P-52, Springer, 2004 (Lecture Notes in Informatics). – ISBN 3–88579–381–4, S. 271–282

[KLM08] Kutz, Oliver ; Lücke, Dominik ; Mossakowski, Till: Modular Construction of Models - Towards a Consistency Proof for the Foundational Ontology DOLCE. In: *First*

International Workshop on Foundations of Computer Science as Logic-Related, (ICTAC-08), 2008

[Koh06] Kohlhase, Michael: OMDoc – *An open markup format for mathematical documents [Version 1.2]*. Springer Verlag, 2006 (LNAI 4180)

[KSV02] Kupferman, Orna ; Sattler, Ulrike ; Vardi, Moshe Y.: The Complexity of the Graded μ-Calculus. In: *Automated Deduction, CADE 02* Bd. 2392, Springer, 2002 (LNCS). – ISBN 3–540–43931–5, 423–437

[LS08] Lukasiewicz, Thomas ; Straccia, Umberto: Managing uncertainty and vagueness in description logics for the Semantic Web. In: *J. Web Sem* 6 (2008), Nr. 4, 291–308. http://dx.doi.org/10.1016/j.websem.2008.04.001

[LS10] Lutz, Carsten ; Schröder, Lutz: Probabilistic Description Logics for Subjective Uncertainty. In: Lin, Fangzhen (Hrsg.) ; Sattler, Ulrike (Hrsg.): *Principles of Knowledge Representation and Reasoning, KR 2010*, AAAI Press, 2010. – To appear

[Luk08] Lukasiewicz, Thomas: Expressive probabilistic description logics. In: *Artif. Intell* 172 (2008), 852–883. http://dx.doi.org/10.1016/j.artint.2007.10.017

[MAH01] Mossakowski, Till ; Autexier, Serge ; Hutter, Dieter: Extending Development Graphs With Hiding. In: Hussmann, Heinrich (Hrsg.): *Proceedings of Fundamental Approaches to Software Engineering (FASE 2001)*. Genova : Springer, April 2001 (LNCS 2029), S. 269–283

[MAH06] Mossakowski, Till ; Autexier, Serge ; Hutter, Dieter: Development Graphs – Proof Management for Structured Specifications. In: *Journal of Logic and Algebraic Programming* 67 (2006), Nr. 1–2, S. 114–145

[MHAH04] Mossakowski, T. ; Hoffman, P. ; Autexier, S. ; Hutter, D.: CASL Logic. In: Mosses, P. (Hrsg.): *The CASL Reference Manual*. Springer-Verlag, LNCS 2960, 2004

[MKB04] Mahnke, Achim ; Krieg-Brückner, Bernd.: Literate Ontology Development. In: Meersman, Robert (Hrsg.) ; Tari, Zahir (Hrsg.) ; al., Angelo C. (Hrsg.): *On the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops* Bd. 3292, Springer, 2004 (Lecture Notes in Computer Science), S. 753–757

[MMi] *MMiSS: Multimedia in Safe and Secure Systems*. Web site at `www.mmiss.de`,

[MML07] Mossakowski, Till ; Maeder, Christian ; Lüttich, Klaus: The Heterogeneous Tool Set. In: Grumberg, Orna (Hrsg.) ; Huth, Michael (Hrsg.): *TACAS 2007* Bd. 4424, Springer-Verlag Heidelberg, 2007 (Lecture Notes in Computer Science), S. 519–522

[Mos00] Mossakowski, Till: Specification in an arbitrary institution with symbols. In: Choppy, C. (Hrsg.) ; Bert, D. (Hrsg.) ; Mosses, P. (Hrsg.): *Recent Trends in Algebraic Development Techniques, 14th International Workshop, WADT'99, Bonas, France* Bd. 1827. Springer-Verlag, 2000, S. 252–270

[Mos05a] Mossakowski, Till: Heterogeneous specification and the heterogeneous tool set / Universitaet Bremen. 2005. – Forschungsbericht. – Habilitation thesis

[Mos05b] Mossakowski, Till: *Heterogeneous Specification and the Heterogeneous Tool Set*, Universität Bremen, Habilitation, 2005

[MPS09] Myers, Rob ; Pattinson, Dirk ; Schröder, Lutz: Coalgebraic Hybrid Logic. In: Alfaro, Luca de (Hrsg.): *Foundations of Software Science and Computation Structures, FoSSaCS 2009* Bd. 5504, Springer, 2009 (LNCS), S. 137–151

[MR07] Mossakowski, Till ; Roggenbach, Markus: Structured CSP – A Process Algebra as an Institution. In: Fiadeiro, J. (Hrsg.): *WADT 2006* Bd. 4409, Springer-Verlag Heidelberg, 2007 (Lecture Notes in Computer Science), S. 92–110

[Neb90] Nebel, Bernhard: Terminological reasoning is inherently intractable. In: *Artificial Intelligence* 43 (1990), S. 235–249

[OMR12] O'Reilly, Liam ; Mossakowski, Till ; Roggenbach, Markus: Compositional modelling and reasoning in an institution for processes and data. In: *Recent Trends in Algebraic Development Techniques*, Springer, 2012 (LNCS)

[PS08] Pattinson, Dirk ; Schröder, Lutz: Beyond Rank 1: Algebraic Semantics and Finite Models for Coalgebraic Logics. In: Amadio, Roberto (Hrsg.): *Foundations of Software Science and Computation Structures, FoSSaCS 2008)* Bd. 4962, Springer, 2008 (LNCS), S. 66–80

[PS09] Pattinson, Dirk ; Schröder, Lutz: Generic Modal Cut Elimination Applied to Conditional Logics. In: Giese, Martin (Hrsg.) ; Waaler, Arild (Hrsg.): *Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX 2009* Bd. 5607, Springer, 2009 (LNCS), S. 280–294

[PS10] Pattinson, Dirk ; Schröder, Lutz: Cut Elimination in Coalgebraic Logics. In: *Information and Computation, :* 208 (2010), S. 1447–1468

[Rog06] Roggenbach, Markus: CSP-CASL: A new integration of process algebra and algebraic specification. In: *Theoretical Computer Science* 354 (2006), Nr. 1, S. 42–71. http://dx.doi.org/http://dx.doi.org/10.1016/j.tcs.2005.11.007. – DOI http://dx.doi.org/10.1016/j.tcs.2005.11.007. – ISSN 0304–3975

[Sch07] Schröder, Lutz: A Finite Model Construction for Coalgebraic Modal Logic. In: *J. Log. Algebr. Prog.* 73 (2007), S. 97–110

[SH02] Schairer, Axel ; Hutter, Dieter: Proof Transformations for Evolutionary Formal Software Development. In: *Proceedings 9th International Conference on Algebraic Methodology And Software Technology, AMAST2002*, Springer-Verlag, LNCS 2422, 2002

[SP07] Schröder, Lutz ; Pattinson, Dirk: Modular algorithms for heterogeneous modal logics. In: *Automata, Languages and Programming, ICALP 07* Bd. 4596, Springer, 2007 (LNCS), S. 459–471

[SP08a] Schröder, Lutz ; Pattinson, Dirk: How Many Toes Do I Have? Parthood and Number Restrictions in Description Logics. In: *Principles of Knowledge Representation and Reasoning, KR 2008*, AAAI Press, 2008, S. 307–218

[SP08b] Schröder, Lutz ; Pattinson, Dirk: Shallow models for non-iterative modal logics. In: *Advances in Artificial Intelligence, KI 2008* Bd. 5243, Springer, 2008 (LNAI), S. 324–331

[SP09a] Schröder, Lutz ; Pattinson, Dirk: PSPACE Bounds for Rank-1 Modal Logics. In: *ACM Trans. Comput. Log.* 10 (2009), S. 13:1–13:33

[SP09b] Schröder, Lutz ; Pattinson, Dirk: Strong completeness of coalgebraic modal logics.

In: Albers, Susanne (Hrsg.) ; Marion, Jean-Yves (Hrsg.): *Symposium on Theoretical Aspects of Computer Science, STACS 2009*, Schloss Dagstuhl – Leibniz-Center of Informatics, 2009 (Leibniz International Proceedings in Informatics), S. 673–684

[SP10a] Schröder, Lutz ; Pattinson, Dirk: Coalgebraic correspondence theory. In: Ong, Luke (Hrsg.): *Foundations of Software Science and Computation Structures, FoSSaCS 2010* Bd. 6014, Springer, 2010 (LNCS), S. 328–342

[SP10b] Schröder, Lutz ; Pattinson, Dirk: Named Models in Coalgebraic Hybrid Logic. In: Marion, Jean-Yves (Hrsg.) ; Schwentick, Thomas (Hrsg.): *Symposium on Theoretical Aspects of Computer Science, STACS 2010* Bd. 5, Schloss Dagstuhl – Leibniz-Center of Informatics, 2010 (Leibniz International Proceedings in Informatics), S. 645–656

[SPH10] Schröder, Lutz ; Pattinson, Dirk ; Hausmann, Daniel: Optimal Tableaux for Conditional Logics with Cautious Monotonicity. In: Wooldridge, Michael (Hrsg.): *European Conference on Artificial Intelligence, ECAI 2010*, IOS Press, 2010 (Frontiers in Artificial Intelligence and Applications). – To appear

[SPK09] Schröder, Lutz ; Pattinson, Dirk ; Kupke, Clemens: Nominals for Everyone. In: Boutilier, Craig (Hrsg.): *International Joint Conferences on Artificial Intelligence, IJCAI 2009*, AAAI Press, 2009, S. 917–922

[SV10] Schröder, Lutz ; Venema, Yde: Flat coalgebraic fixed point logics. In: *CoRR* abs/1004.2717 (2010)

[WMS07] Wölfl, Stefan ; Mossakowski, Till ; Schröder, Lutz: Qualitative Constraint Calculi: Heterogeneous Verification of Composition Tables. In: Wilson, David (Hrsg.) ; Sutcliffe, Geoff (Hrsg.): *20th International FLAIRS Conference (FLAIRS-20)*, AAAI Press, 2007, S. 665–670