jETI: Ein serviceorientiertes Framework zur High-Level Ausführung von Remote-Komponenten

Dissertation

zur Erlangung des Grades eines Doktors der Naturwissenschaften
der Technischen Universität Dortmund
am Fachbereich Informatik

von

Christian Kubczak

Dortmund 2012

Tag der mündlichen Prüfung: 04.01.2013

Dekanin: Prof. Dr. Gabriele Kern-Isberner

Referent: Prof. Dr. Bernhard Steffen, Technische Universität Dortmund
Koreferent: Prof. Dr. Ramin Yahyapour, Georg-August-Universität Göttingen

Erklärung

Ich erkläre hiermit, dass ich die vorgelegte Arbeit selbstständig erstellt und keine anderen als die angegebenen Hilfsmittel und Quellen verwendet habe. Des Weiteren wurden alle Zitate kenntlich gemacht. Referenzierte Internet-Verweise waren zum Zeitpunkt der Erstellung dieser Arbeit in vollem Umfang gültig und erreichbar.
Die für diese kumulativ verfasste Dissertation unmittelbar relevanten Publikationen sind im folgenden aufgeführt und mein Beitrag herausgestellt. Ein komplettes Verzeichnis all meiner Veröffentlichungen findet sich im Anschluss.

Publikationen

Dissertationsrelevante Veröffentlichungen

Konferenzen

1. **Biological LC/MS Preprocessing and Analysis with jABC, jETI and xcms**
   C. Kubczak, T. Margaria, A. Fritsch, B. Steffen
   In *Proc. of 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2006)*, pp. 308-313, Paphos, Cyprus, 15-19 November 2006. IEEE Computer Society Press.

2. **The FMICS-jETI Platform: Status and Perspectives**
   T. Margaria, C. Kubczak, B. Steffen, S. Naujokat
   In *Proc. of 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2006)*, pp. 414-418, Paphos, Cyprus, 15-19 November 2006. IEEE Computer Society Press.

3. **eXtreme Model-Driven Design with jABC**
   C. Kubczak, S. Jörges, T. Margaria, B. Steffen
   In *Vogel, R. (ed.) Proc. of the Tools and Consultancy Track of the Fifth European Conference on Model-Driven Architecture Foundations and Applications (ECMDA-FA), Volume WP09-12 of CTIT proceedings*, P.O. Box 217, 7500 AE Enschede, The Netherlands, University of Twente, Centre for Telematics and Information Technology (CTIT), June 25-26, 2009, pp. 78–99, ISSN: 0929-0672, http://www.ctit.utwente.nl/

4. **Abductive Synthesis of the Mediator Scenario with jABC and GEM**
   C. Kubczak, T. Margaria, M. Kaiser, J. Lemcke, B. Knuth
   In *Proc. of 2008 EON-SWSC Workshop on Evaluation of Ontology-based Tools and the Semantic Web Services Challenge*, Tenerife, Spain, 1-2 June 2008, http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-359/

5. **Synthesizing Semantic Web Service Compositions with jMosel**

and Golog
T. Margaria, D. Meyer, C. Kubczak, M. Isberner, B. Steffen
In *Proc. of the 8th International Semantic Web Conference (ISWC 2009)*, LNCS N.5823/2009, pp. 392–407, Washington D.C., United States, 25-29 October 2009, Springer Verlag.

Fachzeitschriften

6. **Bio-jETI: a Service Integration, Design, and Provisioning Platform for Orchestrated Bioinformatics Processes**
T. Margaria, C. Kubczak, B. Steffen
In *BioMed Central (BMC) Bioinformatics Supplement dedicated to Network Tools and Applications in Biology 2007 Workshop (NETTAB 2007)*, V.9/4 2007, http://www.biomedcentral.com/1471-2105/9?issue=S4

Bücher

7. **Service-oriented Mediation with jABC/jETI**
C. Kubczak, T. Margaria, B. Steffen, R. Nagel
In *Semantic Web Services Challenge - Results from the First Year*, Charles Petrie, Tiziana Margaria, Michal Zaremba, and Holger Lausen (eds.), Springer Verlag, 2008, pp. 71-99, .- ISBN: 978-0-387-72495-9

8. **Automatic Generation of the SWS-Challenge Mediator with jABC/ABC**
T. Margaria, M. Bakera, C. Kubczak, S. Naujokat, B. Steffen
In *Semantic Web Services Challenge - Results from the First Year*, Charles Petrie, Tiziana Margaria, Michal Zaremba, and Holger Lausen (eds.), Springer Verlag, 2008, pp. 119-138, .- ISBN: 978-0-387-72495-9

9. **Comparison: Mediation on WebML/WebRatio and jABC/jETI**
M. Brambilla, S. Ceri, E. Della Valle, F.M. Facca, C. Kubczak, T. Margaria, B. Steffen, C. Winkler
In *Semantic Web Services Challenge - Results from the First Year*, Charles Petrie, Tiziana Margaria, Michal Zaremba, and Holger Lausen (eds.), Springer Verlag, 2008, pp. 153-166, .- ISBN: 978-0-387-72495-9

10. **An Approach to Discovery with miAamics and jABC**
C. Kubczak, T. Margaria, B. Steffen, C. Winkler, H. Hungar
In *Semantic Web Services Challenge - Results from the First Year*, Charles Petrie, Tiziana Margaria, Michal Zaremba, and Holger Lausen (eds.), Springer Verlag, 2008, pp. 217-234, .- ISBN: 978-0-387-72495-9

11. **Comparison: Discovery on WSMOLX and miAamics/jABC**
    C. Kubczak, T. Vitvar, C. Winkler, R. Zaharia, M. Zaremba
    In *Semantic Web Services Challenge - Results from the First Year*, Charles Petrie, Tiziana Margaria, Michal Zaremba, and Holger Lausen (eds.), Springer Verlag, 2008, pp. 247-262, .- ISBN: 978-0-387-72495-9

Meine Beiträge

1. **Biological LC/MS Preprocessing and Analysis with jABC, jETI and xcms**
   Die Arbeit ist in fachlichem Austausch mit dem Lehrstuhl für mathematische Statistik und biometrische Anwendung der Technischen Universität Dortmund entstanden. Ich bin Hauptautor der Abschnitte 1, 3, 4 und 5 und war als Verantwortlicher für das jETI Projekt für die technische Realisierung und die Integration der von den Statistikern gelieferten Auswertungsmethoden zuständig. Die Arbeit wurde von mir im Rahmen der ISoLA 2006 präsentiert.

2. **The FMICS-jETI Platform: Status and Perspectives**
   Die Arbeit ist weitestgehend in Diskussion aller Autoren entstanden. Ich bin Co-Autor aller Abschnitte und war als Verantwortlicher für das jETI Projekt für die technische Durchführung und Integration der Dienste aus dem Bereich der formalen Methoden zuständig. Ein aus dieser Veröffentlichung hervorgegangener Workshop zur Integration weiterer Funktionalitäten aus dem Bereich des Model Checking mit der Universität Malaga, Spanien, wurde von mir durchgeführt.

3. **eXtreme Model-Driven Design with jABC**
   Die Arbeit entstand in ständigem Austausch aller Autoren. Ich bin Hauptautor des Abschnitts 3 und Co-Autor des Abschnitts 4. Die Entwicklung und Ausarbeitung der Fallstudie sowie deren technische Durchführung lag in meinem Verantwortungsbereich. Die Arbeit wurde von mir im Rahmen der ECMDA 2009 präsentiert.

4. **Abductive Synthesis of the Mediator Scenario with jABC and GEM**
   Die Veröffentlichung fand in enger Zusammenarbeit und fachlichem Austausch mit SAP Research, Palo Alto, CA, USA statt. Ich bin Hauptautor der Abschnitte 1 und 4 sowie Co-Autor der Abschnitte 2 und 3. Die technische Realisierung und insbesondere die Integration des von SAP gelieferten Syntheseansatzes wurde dabei von mir durchgeführt.

5. **Synthesizing Semantic Web Service Compositions with jMosel**

and Golog
Die Arbeit entstand in Zusammenarbeit mit dem Lehrstuhl für Software Engineering der Universität Potsdam. Als Verantwortlicher für das jETI Projekt und aktiver Teilnehmer der SWSC war ich insbesondere für die technische Integration der Plan-Dienste und die Lösung des Mediations-Szenarios im Rahmen der Fallstudie zuständig. Ich bin Co-Autor der Abschnitte 1,2 und 5.

6. **Bio-jETI: a Service Integration, Design, and Provisioning Platform for Orchestrated Bioinformatics Processes**
   Als Co-Autor der Arbeit war ich im Wesentlichen für den Anteil der durchgeführten LC/MS Studie, welche detailliert in der ersten Publikation abgehandelt wird, zuständig. Als Verantwortlicher des jETI Projekts oblag mir zudem die technische Realisierung der Bio-jETI Plattform und die Integration weiterer Dienste in diesem Kontext.

7. **Service-oriented Mediation with jABC/jETI**
   Zur Umsetzung des Buchbeitrages standen die Autoren in ständigem Austausch und diskutierten alle Teile der Veröffentlichung miteinander. Die durchgeführte Fallstudie im Rahmen der Semantic Web Services Challenge wurde größtenteils von mir konzeptioniert und umgesetzt. Insbesondere die automatische Integration der Web Services und die Umsetzung auf Basis der jETI Remote Service Umgebung lag in meinem Aufgabenbereich. Ich bin Hauptautor der Abschnitte 5.1 und 5.6 - 5.11 sowie Co-Autor von Abschnitt 5.4. Die erzielten Fortschritte und Lösungen wurden von mir in diversen Workshops im Kontext der SWSC präsentiert.

8. **Automatic Generation of the SWS-Challenge Mediator with jABC/ABC**
   Zur Umsetzung des Buchbeitrages standen die Autoren in ständigem Austausch und diskutierten alle Teile der Veröffentlichung miteinander. Als aktiver Teilnehmer der Arbeitsgruppe an der Semantic Web Services Challenge sowie als Projektverantwortlicher auf Seiten der jETI Integration und Umsetzung war ich als Co-Autor an den Abschnitten 7.1 - 7.4 sowie 7.7 - 7.9 beteiligt. Die erzielten Fortschritte und Lösungen wurden zum Teil von mir in diversen Workshops im Kontext der SWSC präsentiert.

9. **Comparison: Mediation on WebML/WebRatio and jABC/jETI**
   Zur Findung einer Vergleichsbasis und Umsetzung der Veröffentlichung fand ein reger Austausch zwischen den Arbeitsgruppen im Kontext der Semantic Web Services Challenge statt. Insbesondere im Rahmen der technischen Vergleiche war ich als Co-Autor an den Abschnitten 9.3 - 9.6 beteiligt.

10. **An Approach to Discovery with miAamics and jABC**
    Ausgehend von den Arbeiten innerhalb meiner Diplomarbeit [Kub05] war

ich insbesondere mit der Anpassung des Systems an die Gegebenheiten im Rahmen der Semantic Web Services Challenge und der damit verbundenen Lösung des Discovery-Szenarios beschäftigt. Zur Umsetzung der Arbeit fand zudem ein Austausch zwischen allen beteiligten Autoren statt. Ich bin Hauptautor des Abschnitts 2 und Co-Autor der Abschnitte 1, 3 und 4.

11. **Comparison: Discovery on WSMOLX and miAamics/jABC**
    Neben der ständigen Diskussion aller beteiligten Autoren und insbesondere zwischen den Arbeitsgruppen im Rahmen der Semantic Web Services Challenge bin ich im Zuge des technischen Vergleichs der Systeme Hauptautor der Abschnitte 1 und 4, sowie Co-Autor der Abschnitte 2 und 3.

Weitere Publikationen

Diplomarbeit

12. **Entwicklung einer verteilten Umgebung zur Personalisierung von Web-Applikationen**
    C. Kubczak.
    Technische Universität Dortmund, 2005

Konferenzen

13. **Model-Driven Development with the jABC**
    B. Steffen, S. Jörges, R. Nagel, C. Kubczak, T. Margaria
    In *Hardware and Software, Verification and Testing: Proc. of 2006 IBM Haifa Verification Conference (HVC 2006)*, LNCS N.4383/2007, pp. 92-108, Haifa, Israel, 23-26 October 2006. Springer Verlag.

14. **Model-based Design of Distributed Collaborative Bioinformatics Processes in the jABC**
    T. Margaria, C. Kubczak, M. Njoku, B. Steffen
    In *Proc. of 11th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2006)*, pp. 169-176, Stanford, California, USA, 15-17 August 2006. IEEE Computer Society Press.

15. **The SWS Mediator with WebML/WebRatio and jABC/jETI: A Comparison**
    T. Margaria, C. Winkler, C. Kubczak, B. Steffen, M. Brambilla, S. Ceri, D. Cerizaa, E. Della Valle, F. Facca, C. Tziviskou
    In *Proc. of 9th International Conference on Enterprise Information Systems*

(ICEIS 2007), pp. 422-429, Funchal, Madeira, Portugal, 12-16 June 2007.

16. **Service-oriented Mediation with jETI/jABC: Verification and Export**
    C. Kubczak, T. Margaria, B. Steffen, S. Naujokat
    In *Proc. of 2007 International Conference on Web Intelligence and Intelligent Agent Technology (IEEE/WIC/ACM 2007) – WI-IAT Workshop*, pp. 144-147, Silicon Valley, California, USA, 2-5 November 2007. IEEE Computer Society Press.

17. **An approach to Discovery with miAamics and jABC**
    C. Kubczak, T. Margaria, C. Winkler, B. Steffen
    In *Proc. of 2007 International Conference on Web Intelligence and Intelligent Agent Technology (IEEE/WIC/ACM 2007) - WI-IAT Workshop*, pp. 157-160, Silicon Valley, California, USA, 2-5 November 2007. IEEE Computer Society Press.

18. **Model Driven Design of Reliable Robot Control Programs Using the jABC**
    S. Jörges, C. Kubczak, F. Pageau, T. Margaria
    In *Proc. of 4th IEEE International Workshop on Engineering of Autonomic and Autonomous Systems (EASe 2007)*, pp. 137-148, Tucson, Arizona, USA, 26-29 March 2007. IEEE Computer Society Press.

Fachzeitschriften

19. **Evolution support in heterogeneous service-oriented landscapes**
    T. Margaria and B. Steffen and C. Kubczak
    In *In Journal of the Brazilian Computer Society*, Springer London, 2010, V.16/1, pp. 35-47, .- ISSN: 0104-6500

Bücher

20. **The XMDD Approach to the Semantic Web Services Challenge**
    T. Margaria and C. Kubczak and B. Steffen
    In *Semantic Web Services – Advancement through Evaluation*, M.B. Blake, L. Cabral, B. König-Ries, U. Küster, and D. Martin (eds.), Springer Verlag, 2012, pp. 233-248, .- ISBN: 978-3-642-28734-3

Technische Berichte

21. **Abductive Synthesis of the Mediator Scenario with jABC and GEM**
C. Kubczak, T. Margaria, M. Kaiser, J. Lemcke, B. Knuth
In *Semantic Web Services Challenge: Proceedings of the 2008 Workshops*, Charles Petrie (ed.), Stanford University, Stanford, CA, 2009, LG-2009-01, pp. 52-63. http://logic.stanford.edu/reports/LG-2009-01.pdf

22. **Advances in Solving the Mediator Scenario with jABC and jABC/GEM**
J. Lemcke, M. Kaiser, C. Kubczak, T. Margaria, B. Knuth
In *Semantic Web Services Challenge: Proceedings of the 2008 Workshops*, Charles Petrie (ed.), Stanford University, Stanford, CA, 2009, LG-2009-01, pp. 89-102. http://logic.stanford.edu/reports/LG-2009-01.pdf

Industrierelevante Veröffentlichungen

23. **Geballte Kompetenz – Integration und verteilte Nutzung von entfernten Diensten zur Prozessmodellierung**
C. Kubczak, T. Margaria, B. Steffen, P. Grünewald
In *Biotechnologie Special (BioTechnica 2007)*, Verfahrenstechnik N.10/2007, pp. 2-3.

24. **Plug and Play with FMICS-jETI: Beyond Scripting and Coding**
C. Kubczak, T. Margaria, R. Nagel, B. Steffen
In *ERCIM News*, N.73, April 2008, pp. 41-42. http://www.biomedcentral.com/1471-2105/9?issue=S4.

Für Ben.

Danksagung

Viele Menschen haben mich während der letzten Jahre, in denen diese Arbeit entstand, motiviert, angetrieben und inspiriert.
Mein Dank gilt zunächst meinem Doktorvater Prof. Dr. Bernhard Steffen für seine konstante Unterstützung, Hartnäckigkeit und Geduld, sowie den beständigen Glauben an mich und die Möglichkeit, in seiner Arbeitsgruppe tätig gewesen zu sein.
Desweiteren danke ich Prof. Dr.-Ing. Ramin Yahyapour für die Bereitschaft, die Koreferentschaft für diese Arbeit zu übernehmen, meinem Kollegen Dr. Sven Jörges für die schnelle und gründliche Korrekturlesung, Stefan Naujokat, der vor und während seiner Diplomarbeit maßgeblich an der technischen Umsetzung der jETI Plattform mitgewirkt hat und Anna-Lena Lamprecht für Ihre Unterstützung im Rahmen des Bio-jETI Projekts.
Spezieller Dank gilt Dr. Nikolas Funke für die Motivation in schweren Stunden, ein schwarzes Buch und eine Unterkunft an Englands Küste. Ines Maybaum, Andreas Brügge, Ronald Hübner und Sammy Amara danke ich für die Musik, die Geduld und die Ablenkung, wenn die Arbeit wieder einmal Überhand nahm.
All meinen Kollegen bei der e-Spirit AG danke ich für die Unterstützung und die Schaffung der Freiräume, die ein Fertigstellen dieser Arbeit erst möglich gemacht haben.
Bei Prof. Dr.-Ing. Tiziana Margaria möchte ich mich ausdrücklich für die tolle Zusammenarbeit, ihre Sicht der Dinge und all die Arbeitsgruppen bedanken, an denen ich teilhaben durfte. An dieser Stelle ist das gesamte Team der Semantic Web Services Challenge besonders hervorzuheben.
Zu guter Letzt möchte ich mich bei meiner Familie und insbesondere bei meiner Frau Miriam für die Unterstützung, die Geduld und den Zuspruch bedanken, der mir immer wieder neue Kraft beim Verfassen dieser Arbeit gegeben hat.
Dr. Gabriele Götz und Dr. Klaus Schwartmann danke ich für den geweckten Ehrgeiz. Ich werde Euch nie vergessen.

Zusammenfassung

Mit dem massiven Aufkommen an geschäftlichem Datenaustausch in Zeiten voranschreitender Globalisierung, dem Erfolg des Community getriebenen Web 2.0 und dem expliziten Wandel hin zur serviceorientierten Denkweise insbesondere im Kontext komplexer Geschäftsprozesse wird vor allem deutlich, dass wachsender Bedarf an einer offenen, standardisierten Technologie zur Vernetzung, Ausführung und Bereitstellung von Diensten und Prozessen besteht.
Um diesen Bedarf zu decken, bestehende Probleme zu lösen und neue Aspekte der Ausführung vernetzter Dienste zu erkennen soll in dieser Arbeit eine ganzheitliche Experimentierplattform zur serviceorientierten Orchestrierung, insbesondere von verteilt agierenden Komponenten vorgestellt werden. Nutzer sollen dabei in die Lage versetzt werden, Funktionalitäten auf eine möglichst einfache Art und Weise ausprobieren, benutzen und anbieten zu können. Im Kern soll hier der Teilbereich einer integrierten Umgebung beleuchtet werden, welche die Verifikation und Handhabung von standardisierten Remote Service Komponenten als auch die Anbindung heterogener, proprietärer Dienste von Drittanbietern unterstützt. Die vorgelegte Arbeit umfasst dabei vor allem die Konzeption und Realisierung einer ganzheitlichen Service-Integrationsplattform, welche es nicht ausschließlich, aber insbesondere Domänenexperten ermöglichen soll, Funktionalitäten verteilt und effizient anbieten und nutzen zu können, was anhand komplexer und divergenter Fallstudien aus verschiedenen Fachrichtungen belegt wird. Zum Testen und evaluieren komplexer Technologien soll dabei zusätzlich eine integrierte Testumgebung dienen. Aktuelle Technologien des Semantic Web vervollständigen letztendlich das Einsatzgebiet in puncto automatischer Modellierung und Verifikation sowie dynamischer Suche von adäquat dedizierten Diensten.

Inhaltsverzeichnis

Teil I.

Grundlagen und Motivation

1. Einführung

Serviceorientierte Architekturen (SOA) [Joa11] und der modellgetriebene Ansatz (MDSD) [MS06, MS08, MS04] zur Entwicklung komplexer Applikationen auf Basis von Prozessmodellen sind heute in Ihrer Gänze gerade für die Zielgruppen, die am meisten davon profitieren könnten weithin nur partiell erfassbar. Dies umfasst insbesondere die Gruppe der Applikations-/Domänenexperten, Prozessmodellierer und Personenkreise mit wenig Anwendungswissen aus dem Bereich der Softwareentwicklung. Hier könnte unter Berücksichtigung von bereitgestellten und vordefinierten Diensten für explizite Problemstellungen den Anwendern in besonderem Maße ein Werkzeug an die Hand gegeben werden, mit dessen Hilfe es Ihnen ermöglicht würde, komplexe Lösungen für domänenspezifische Aufgabenstellungen selbst zu erarbeiten und konkrete Technologien einfach und effizient zu evaluieren.
Vergleicht man den Prozess der technologischen Entscheidungsfindung und Erarbeitung einer passgenauen Lösung beispielsweise mit der Anschaffung eines Pkw, so existieren in der Realität eine Reihe von Anbietern mit fertigen Lösungen für allgemeine bis hin zu sehr spezifischen Anforderungen. Einfach gehaltene Fahrzeugmodelle mit geringem Wartungsaufwand stehen hier komplexen Fahrzeugen mit einem Höchstmaß an Komfort gegenüber. Extreme Geländemaschinen decken andere Einsatzgebiete ab als Sportwagen. Allen Lösungen ist jedoch gemein, dass sie je nach Preisklasse auf mehr oder weniger individuellen Wunsch des Kunden von entsprechenden Experten ausgestattet und erweitert oder gar komplett von Grund auf gefertigt werden. Hierzu kann auf einen Fundus von Einzelteilen oder Baugruppen verschiedener Hersteller und Drittanbieter zurückgegriffen werden. Technisch versierte Kunden können u.U. auf Bausätze zurückgreifen, benötigen aber ein hohes Maß an technischem Know-how und entsprechenden Fertigkeiten sowie die Infrastruktur zum Bau eines Fahrzeugs. Gravierende Unterschiede stellen sich zudem ein, wenn es zum Prozess der einfachen Evaluierung von Systemkomponenten, ähnlich der Probefahrt eines Pkw, kommt. Hier fehlen der Softwareindustrie i.d.R. adäquate Methoden, um diesen Aspekt kostengünstig abzudecken. Ein langwieriger und aufgrund dessen oftmals kostspieliger Austausch von Anforderungen, Spezifikationen und konkretem Fachwissen mit IT-Beratern und Anwendungsentwicklern ist die Folge. Gleiches gilt für die Anbieter von spezifischen Diensten, die ihrerseits ebenfalls auf vorhandene Services und entsprechende Infrastruktur zurückgreifen müssen, um Funktionalitäten zu entwickeln oder zu erweitern und für die Nutzung innerhalb einer SOA bereit-

Abbildung 1.1.: Lösungsprozess zwischen Expertenwissen und Kosten

zustellen. Neben der Tatsache, dass SOA Implementierungen leider oftmals sehr technisch umgesetzt werden, gibt es außerdem eine ganze Reihe von divergenten Umsetzungen des Konzepts. Dies führt i.d.R. insbesondere dazu, dass an den o.g. wichtigen Benutzergruppen vorbeientwickelt wird und lediglich der Wissensschwerpunkt von IT-Experten weg von der klassischen Programmierung hin zu einer Service-gestützen Entwicklung verlagert wird. Um das obige Beispiel der Automobilindustrie aufzugreifen, muss ein Kfz Mechaniker heutzutage neben den grundlegenden Fertigkeiten völlig anderen Anforderungen genügen, als es noch vor wenigen Jahren der Fall war. Modulare Konzepte haben Einzug in den modernen Fahrzeugbau gehalten, erschließen sich aber dennoch nur einem sehr speziellen Personenkreis. Reparaturen und Wartungsarbeiten werden zwar durch den Austausch industriell vorgefertigter Komponenten und Baugruppen verschiedenster Anbieter vereinfacht, erfordern aber nach wie vor einen entsprechend ausgebildeten Techniker. Das technische Wissen wird also lediglich innerhalb eines abgeschlossenen Expertenkreises auf eine neue Abstraktionsebene gehoben.
Als softwaretechnisches Beispiel einer ursprünglich guten Idee, welche unter einer ausufernden Umsetzung zu leiden hat seien hier vorab die sogenannten *Web Services* genannt (siehe Abschnitt 3). Diese stellen konzeptuell ein mächtiges

Werkzeug für den Austausch heterogener Funktionalitäten und Technologien dar und gelten de facto als **der** Standard einer verteilten, wiederverwendbaren, agilen und validen Service Komponente. In der Realität ist es jedoch gerade aufgrund der hohen Mächtigkeit von Web Services so, dass die Implementierung von Funktionalitäten auf der einen und die Nutzung der angebotenen Dienstleistungen auf der anderen Seite ein hohes Maß an technischem Verständnis und Erfahrung erfordert. Kernfunktionalitäten müssen letztendlich über Benutzerschnittstellen von entsprechenden Entwicklungsexperten an den Endnutzer weitergegeben werden, damit diese effizient von den Diensten profitieren können. Die Vorteile der Serviceorientierung für technisch unversierte Nutzer werden dadurch aber zum Teil komplett verdeckt, da es sich im Grunde um eine klassische Nutzung einer Softwarekomponente mit Hilfe einer z.B. grafischen Benutzeroberfläche handelt, was prinzipiell der Nutzung einer Desktop- oder Web-Applikation gleich kommt. In der Praxis ist es tatsächlich so, dass für die konkreten Anwender häufig jeglicher Vorteil von Web Services komplett verdeckt wird und die Nutzer keinerlei Unterschied zu einer klassischen Applikation erkennen können.Auf den ersten Blick disqualifiziert sich somit diese Technologie, wenn es um Einsatzbereiche geht, in denen nicht-ITler ohne tiefere Vorkenntnisse solcherlei Dienste, u.U. sogar automatisiert, in komplexe Prozessabläufe integrieren wollen oder müssen (vgl. Fallstudie 6.1).

1.1. Anforderungen

Der einfache und effiziente Einsatz verteilter, externer und heterogener Dienste verschiedener Anbieter besitzt im Bereich der modellgetriebenen Entwicklung eine hohe Relevanz, da auf diese Art und Weise hochspezialisiertes Know-how einzelner Dienstleister einer möglichst großen Zahl an Benutzern mit unterschiedlichen Wissensschwerpunkten zugänglich gemacht werden kann. Ebenso können sehr spezifische oder rechenintensive Anforderungen (z.B. aus dem Bereich der Bioinformatik) auf extrem leistungsfähige und passgenau zugeschnittene Hardware ausgelagert werden. Zusammenfassend lässt sich somit sagen, dass

1. eine serviceorientierte Architektur und die modellgetriebene Entwicklung erst durch die Bereitstellung einer möglichst umfangreichen Bibliothek an Diensten ihre Vorteile voll ausspielen kann und in der Praxis für den Anwender nutzbar wird.

2. bezogen auf Punkt 1 die technologische Komplexität für die Anbieter und Nutzer von Service Komponenten so gering wie möglich gehalten werden muss, um eine große Anzahl heterogener Dienste mit einer hohen Akzeptanz und daraus resultierend einer hohen Verbreitung zu erreichen. Und letztendlich

3. der Bedarf an einer ganzheitlichen Umgebung zur Bereitstellung und Nutzung verteilter Dienste besteht. Integriert in eine entsprechende Modellierungsumgebung ergänzen sich beide Teile zu einem mächtigen Gesamtsystem mit einer hohen Praxisrelevanz im Bereich der serviceorientierten Softwarentwicklung.

Da im Bereich der grafischen Modellierung von Prozessen bereits eine Vielzahl von Werkzeugen und teilweise standardisierte Vorgehensweisen existieren (siehe Kapitel 2), konzentriert sich diese Arbeit auf die praktische Integration von externen Diensten unterschiedlicher Drittanbieter. Um gerade das weite und inhomogene Feld der verteilten Bereitstellung und Ausführung von Service Komponenten, detailliert beschrieben in Abschnitt 3, für Benutzer greifbar zu machen, soll im Kern dieser Arbeit eine möglichst einfache und intuitive Methodik zur serviceorientierten Verwendung von heterogenen, externen Dienstleistungen durch Applikations- und Domänenexperten erarbeitet werden. Dabei kommt es durch das Entwickeln und Bereitstellen eines einfachen, ganzheitlichen Frameworks zur Behandlung von Remote Service Komponenten zu klaren Abgrenzungen gegenüber weiten Teilen der Standardtechnologien (siehe Abschnitt 5).
In Hinblick auf dieses Profil ergeben sich eine Reihe grundlegender Anforderungen an eine entsprechende Umgebung:

A1 - Modularität/Kompositionalität Eine flexible und modulare Dienstanbindung ist unumgänglich, wenn es um den Einsatz heterogener Komponenten externer Anbieter geht. Hier stellt die einheitliche Kapselung der angebotenen Dienste eine Herausforderung dar. Über eine Abstraktionsschicht muss die Dienstschnittstelle auf Seiten der Nutzer und Service Provider auf den kleinsten gemeinsamen Nenner gebracht werden um komplexe Technologien ebenso wie triviale Dienstaufrufe zu normieren. Zudem sollten die zu integrierenden Services möglichst lose miteinander gekoppelt sein, um einen einfachen und agilen Austausch von Komponenten zu realisieren sowie einen produktiven Umgang mit den Diensten auch im Falle eines Verbindungsabbruchs zu gewährleisten. Neben der syntaktischen Modularität von Diensten und Erweiterungen spielt die semantische Kompositionalität ebenfalls eine signifikante Rolle. Dienste müssen logisch korrekt miteinander in Kombination gebracht und diese Orchestrierungen effizient validiert werden können (vgl. A7).

A2 - Universalität Eine Unterstützung von standardisierten und am Markt etablierten Technologien zur Integration verteilter Dienste gewährleistet eine hohe Akzeptanz und reduziert die Hürde, wenn es um die Verbreitung der Technologie und das Erschließen verschiedener Benutzerkreise mit unterschiedlichen technologischen Schwerpunkten geht. Aus diesem Grund muss eine Evaluierung von Standardtechnologien erfolgen und ein Mindestmaß an in verschiedenen Domä-

nen akzeptierten Technologien berücksichtigt werden.

A3 - Monitoring/Benchmarking Um verlässliche Aussagen über das Laufzeitverhalten komplexer Prozesse treffen zu können und Service Komponenten insbesondere daraufhin zu evaluieren (vgl. T6) , muss die Möglichkeit bestehen, Prozessausführungen zu beobachten und qualifiziert bzgl. ihrer Leistungen zu vergleichen.

A4 - Versionierung auf Modellebene In verschiedenen Domänen wie z.B. im Kontext biologischer oder chemischer Experimente spielt die Wiederholung von Prozessabläufen eine wichtige Rolle. U.U. müssen dabei einzelne Parameter und Kontrollflussaspekte der Ausführung verändert werden können. Technisch gesehen muss somit die Möglichkeit einer Versionierung von Prozessmodellen und das Speichern verschiedener Ausführungsabläufe gegeben sein.

A5 - Plattformunabhängigkeit In heterogenen Unternehmenslandschaften muss ein einfacher Zugriff auf das System gewährleistet sein. Unterschiedliche Hardwarekomponenten und dazugehörige Betriebssysteme erfordern ein hohes Maß an Flexibilität. Insbesondere bei der verteilten Verfügbarkeit und entsprechender Nutzung von Service Komponenten muss die Plattformunabhängigkeit der zu entwickelnden Remote Umgebung sichergestellt sein.

A6 - Erweiterbarkeit Eine einheitliche und gut strukturierte Erweiterbarkeit des Systems gewährleistet eine zukunftssichere Entwicklung. Neue Technologien und zusätzliche Funktionalitäten müssen einfach und effizient zu integrieren sein. Insbesondere auf Seiten der Kommunikationsprotokolle unterschiedlicher Remote Ansätze muss eine offene Schnittstelle vorhanden sein, um zusätzliche Diensttechnologien auch in Zukunft einfach integrieren zu können.

A7 - Verifikation Modellierte Systeme und Prozesse unterliegen einem agilen Entwicklungsprozess und müssen definierten Spezifikationen genügen. Eine effiziente und umfangreiche Verifikation von Anforderungen an ein Modell muss gewährleistet sein, um komplexe Strukturen analysieren und überprüfen zu können. Insbesondere hinsichtlich der Kompositionalität (vgl. A1), automatisierter Prozesssynthese (vgl. T1) und Anwendungen im Kontext des Semantic Web (vgl. T3) ist der Umgang mit logischen Formalismen und Regelwerken gefordert.

1.1.1. Spezialisierte Anforderungen

Unter Berücksichtigung der zuvor in Abschnitt 1.1 definierten, grundlegenden Anforderungen ergeben sich im Kontext des Lösens konkreter und praxisrelevanter Problemstellungen aus verschiedenen Anwendungsdomänen weitere Aspekte welche im Zuge von empirischen Fallstudien aufgetreten sind und auch durch diese im weiteren Verlauf der Arbeit belegt werden. Darüber hinaus erfordert die Ausrichtung auf die Zielgruppe der Anwendungs- und Domänenexperten weiterführende Arbeiten.

S1 - Service Level Agreement Dienstanbieter und -nutzer müssen in der Lage sein, die Qualität der zur Verfügung gestellten Dienste (Quality of Service, QOS) auf Basis einer Vereinbarung (Service Level Agreement, SLA) auszuhandeln. Insbesondere spielen Aspekte wie Lizenzmanagement oder die abgesicherte Übertragung von Daten und Ausführung von Diensten eine Rolle. Der Dienstanbieter kann dabei z.B. eine Abstufung der von ihm zur Verfügung gestellten Funktionalitäten vornehmen und so, gerade im betriebswirtschaftlichen und kommerziellen Umfeld, die Produktivität eines Service steigern. Zur abgesicherten Kommunikation bei der Ausführung von verteilten Diensten müssen zudem sensible Daten geschützt werden. Dies spielt insbesondere in industriell motivierten Anwendungen eine tragende Rolle und soll durch den Einsatz gängiger Verschlüsselungsverfahren realisiert werden. Wenn es um das domänenspezifische Bereitstellen von Know-how oder Technologien in Form externer Service Komponenten geht, ist ebenfalls gerade im Bereich der freien Industrie eine Funktion zur wirtschaftlichen Verwertung maßgebend. Die Remote Umgebung muss im Stande sein, genutzte Funktionalitäten verschiedener Anbieter nach unterschiedlichen Kriterien abzurechnen (Billing) oder zu beschränken (Restriktion). Eine entsprechende SLA Komponente soll darüberhinaus alle getroffenen Vereinbarungen zwischen Anbieter und Nutzer verbindlich absichern.

S2 - Dienstintegration Den Grundsätzen einer SOA, bzw. denen des MDSD folgend muss die Integration von verteilten Diensten für den Nutzer intuitiv und ohne technische Grundkenntnisse bzgl. der eingesetzten Technologien erfolgen können. Eine weitestgehend automatisierte Integration möglichst aller eingesetzten technologischen Ansätze von Remote Services wäre somit wünschenswert. Auch hier spielt wie in Anforderung A1 die Abstraktion der Dienstschnittstellen und -funktionalitäten eine wesentliche Rolle. Ein Trade-off zwischen der Unterstützung einer möglichst weitreichenden Anzahl von Dienstfunktionalitäten und der Möglichkeit einer automatisierten Integration ist die Folge.

S3 - Bereitstellung von Services Analog zu S2 soll auch das Bereitstellen von Dienstleistungen möglichst ohne technikrelevante Fachkenntnisse erfolgen können und somit den Anforderungen domänenspezifischer Dienstanbieter genügen. Hier muss für individuelle Domänen die Unterstützung der relevanten Kernfunktionalitäten eines Dienstes unter Berücksichtigung einer weitestgehend vereinfachten und automatisierten Verfügbarmachung sichergestellt werden.

S4 - Discovery Das dynamische und möglichst autonome Auffinden von adäquaten Komponenten aus der Masse der vorhandenen und bereitgestellten Dienste muss auf Basis spezifizierter Anforderungen sichergestellt sein. Unter Berücksichtigung einer formalen Spezifikation muss das System in der Lage sein, wissensbasiert und effizient entsprechende Service Komponenten zu identifizieren und anzubieten. Dies soll mit Hilfe etablierter Verfahren wie beispielsweise dem Reasoning auf Ontologien oder dem effizienten Auswerten komplexer Regelstrukturen auf Basis von algebraischen Entscheidungsdiagrammen (Algebraic Decision Diagrams, ADD) durchgeführt werden.

1.1.2. Technische Anforderungen

Neben den generellen Anforderungen hinsichtlich des Einsatzes und der Benutzbarkeit einer verteilten Remote Service Umgebung existieren eine Reihe technischer Anforderungen, die sich aus der Verwendung verteilter Komponenten im aktuellen technologischen Kontext ergeben. Natürlich wirken diese sich ebenso unmittelbar auf die konkrete Nutzung aus. Bedingt werden sie allerdings aufgrund technologischer Konzepte und Einsatzgebiete.

T1 - Automatisierung In Bezug zu A1 und S2/S3 muss eine Möglichkeit gefunden werden, aktuelle Remote Technologien effizient und möglichst automatisiert zu integrieren. Neben dem Austausch der Protokollschicht bei der Kommunikation zwischen Nutzer und angebotenem Dienst muss evaluiert werden, inwiefern verschiedene Remote Technologien strukturierte Informationen und/oder Schnittstellen zur automatisierten Integration und somit Generierung von ausführbarem Programmcode auf Seiten der Dienstanbieter und -nutzer bereitstellen. Gegebenenfalls muss neben der vollautomatischen Integration ein Trade-off zwischen der Realisierung und der Nutzerfreundlichkeit z.B. in Form von Wizard gestützter Dienstanbindung gefunden werden.
Eine weitere Ebene der Automatisierung ergibt sich durch die Synthese von neuen Submodellen auf Basis bereits integrierter Dienstkomponenten mit Hilfe entsprechender Annotationen. Auf diese Weise sollen datentypgerechte Ein- und Ausgabeoperationen eines Dienstes dem jeweiligen Problemszenario angepasst werden

können. Dies spielt insbesondere bei der Kapselung von Remote Diensten mit komplexen Schnittstellen eine Rolle, um eine größtmögliche Vereinfachung der Parameter auf einer Ein- und Ausgabeseite zu gewährleisten ohne die Flexibilität und Funktionsvielfalt des Dienstes zu beeinflussen. Hier können Dienstkomponenten zur Abstraktion der Schnittstellen eingesetzt und automatisiert mit den entsprechenden Services kombiniert werden. Auch komplexe Prozessmodelle zur Erfüllung bestimmter Dienstkriterien, z.B. bei der Mediation von Daten- und Kontrollfluss, im Kontext des Semantic Web sind denkbar. Eine konsistente Möglichkeit zur Annotation von Dienstbeschreibungen sowie eine Systemkomponente zur Synthese von Prozessmodellen auf Basis formal beschriebener Spezifikationen ist daher notwendig.

T2 - Langlebigkeit Im Bereich der verteilten Ausführung unterschiedlicher Dienste können verschiedene Faktoren zu einer langen Laufzeit der Prozesse beitragen. Langsame Kommunikationsverbindungen zwischen dem Nutzer und dem auszuführenden Dienst, große zu übertragenden Datenmengen oder komplexe Berechnungen mit entsprechend langen Laufzeiten (bei intensiven Rechenprozessen mitunter über Wochen!) können beispielsweise die synchrone Kommunikation der Teilnehmer gefährden oder gar blockieren. Ein flexibles und effizientes Konzept zur asynchronen Kommunikation zwischen verteilten Komponenten spielt somit eine tragende Rolle bei der Umsetzung der Remote Service Umgebung.

T3 - Semantik Das Voranschreiten des Semantic Web und der damit verbundenen Annotation semantischer Informationen an entsprechende Service Komponenten erfordert eine entsprechende konsistente technologische Unterstützung auf Seiten der Remote Umgebung. Insbesondere bei der automatischen Integration von Diensten oder Synthese von Prozessmodellen im Sinne von T1 müssen Informationen über die Komponenten erfasst und adäquat verarbeitet werden. Die Einhaltung und Unterstützung gängiger Standards sowie das der Zielgruppe von Anwendungs- und Domänenexperten entsprechende Aufbereiten dieser Informationen ist dabei ausschlaggebend.

T4 - Hands-On Evaluation Neben den Grundfunktionalitäten um messbare Vergleichswerte von Prozessausführungen zu erzielen und Aussagen über die Leistungsfähigkeit von Komponenten treffen zu können (siehe A3), wird eine einfache und intuitive Methode zum Testen des Umgangs mit den Diensten benötigt. Dabei sollte ein technischer Prüfstand (Testbed) in Form einer verteilten Evaluations- und Experimentierplattform in die Remote Umgebung integriert sein, mit dessen Hilfe die Anwender verschiedene Komponenten in der praktischen Anwendung einfach, isoliert und ohne technischen Installations- oder Integrationsaufwand evaluieren können.

T5 - Ausführbarkeit von Modellen Die zu realisierende Integration von konkreten Remote Technologien führt dazu, dass die erzeugten Modelle in der Lage sein müssen, mit den verteilten Diensten zu kommunizieren und sie innerhalb einer Prozessdefinition auszuführen.

1.2. Meine Arbeitsschwerpunkte

Gemäß der im vorherigen Abschnitt 1.1 definierten Anforderungen und nach Evaluation diverser Frameworks zur modellgetriebenen Entwicklung von Prozessen (siehe dazu Abschnitt 2) entschied ich mich, als grundlegende Technologie für die erarbeitete Remote Service Umgebung das jABC[1] [SMN+06, MS06, Nag09] Modellierungs-Framework einzusetzen, welches in Abschnitt 4 beschrieben wird. Meine Entscheidung wurde hierbei maßgeblich durch die Tatsache beeinflusst, das die o.a. Anforderungen sich ohne Berücksichtigung von Remote Technologien in weiten Teilen mit denen an eine gänzlich lokal arbeitenden Modellierungsumgebung decken und das jABC in meinem unmittelbaren Arbeitsumfeld exakt auf diese Anforderungen hin konzipiert und entwickelt wurde. So werden die Anforderungen A1, A2 und T2 direkt in der Dissertation von R. Nagel [Nag09] abgehandelt. Durch den Einsatz verfügbarer Plugins für das jABC sowie die Implementierung entsprechender Schnittstellen zur Nutzung etablierter Technologien konnte ich zudem die Ausführbarkeit von Modellen (T5),die Verarbeitung von semantischen Informationen (T3) sowie das Monitoring und Benchmarking (A3) abdecken. Eine detaillierte Darstellung meiner Entscheidungsfindung im Rahmen eines Vergleichs aktuell relevanter Modellierungsumgebungen beinhaltet Abschnitt 2. Auf Basis der einzusetzenden Basistechnologie fokussiert meine Arbeit somit die folgenden Teilbereiche:

1. **Architektur der Remote Service Umgebung:** Ausgehend von den Konzepten des serviceorientierten Ansatzes [Joa11] und dem Grundgedanke des eXtreme Model-Driven Design [MS08] „komplex für wenige - einfach für viele“ lag das Hauptproblem zunächst in der Konzeption und Architektur einer Remote Service Umgebung, die diesen Ansprüchen unter Berücksichtigung der im vorherigen Abschnitt definierten Anforderungen gerecht wird. Der Hauptanspruch lag hierbei konsequent auf der Schaffung eines Systems, welches sowohl die Bedürfnisse der Dienstanbieter als auch die der Dienstnutzer berücksichtigt und in einer intuitiven Art und Weise und ohne vorausgehende Programmierkenntnisse eingesetzt werden kann.
   Basierend auf dem jABC als Modellierungsumgebung und ausgehend von einer bereits in einer frühen Version existierenden Integrationsumgebung für verteilte Dienste wurde von mir den Anforderungen folgend eine eigenstän-

[1] `http://www.jabc.de`

dige Client-Server Umgebung geschaffen, welche die einfache Integration sowie die Bereitstellung von externen und dabei insbesondere heterogenen Services ermöglicht. Adäquate Verfahren auf Basis etablierter Technologien zur Realisierung von Langzeitausführungen im Sinne asynchroner Servicekommunikation sowie die Annotation und Aufbereitung semantischer Informationen im Kontext automatisierter Prozesssynthese und Service Discovery spielten dabei eine zentrale Rolle. Ebenso wurde eine abgesicherte Kommunikationsschnittstelle zwischen den Dienstteilnehmern und ein effizientes Verfahren zum Umgang mit Restriktionen und industrieller Wertschöpfung in Form von Lizenzpolitik integriert. Der genaue Aufbau meiner Remote Service Umgebung hinsichtlich der im vorherigen Abschnitt definierten Anforderungen sowohl auf konzeptueller als auch technischer Ebene ist in Abschnitt 5 beschrieben.

2. **Automatisierte Integration und Komposition:** Neben der eigenständig entwickelten Remote Umgebung, welche sich zwar im besonderen auf im jABC modellierte Prozesse bezieht, aber im allgemeinen losgelöst von der Modellierungsumgebung und somit autonom arbeitet, habe ich eine Integrationskomponente für Standardtechnologien im Kontext verteilter Dienste und Applikation entwickelt. Diese wird eng an die grafische Modellierungsumgebung gekoppelt und dient im Wesentlichen der Unterstützung gängiger Standards. Als Plugin realisiert, lässt sie sich nahtlos in das jABC integrieren und bietet eine weite Unterstützung von relevanten Technologien, wie z.B. Web Services, REST konformen Diensten oder speziell angepassten Schnittstellen wichtiger Werkzeuge (z.B. SAP, R-Project, siehe Abschnitt 6.4 und 6.1). Den Integrationsprozess der Dienstkomponenten habe ich dabei auf Basis struktureller Schnittstellenbeschreibungen sowie semantischen Annotationen so weit als möglich automatisiert, was es insbesondere Anwendungs- und Domänenexperten ohne Programmierkenntnisse erlauben soll, diese in weiten Teilen der Industrie und Wissenschaft eingesetzten Methoden eigenständig zu verwenden (vgl. Abschnitt 5.2.2). Mein Anspruch lag somit über den gesamten Arbeitszeitraum auf der Realisierung einer Abstraktionsebene, welche die technischen Details komplett vor den Nutzern verbirgt und eine möglichst einfache Verwendung aktueller und heterogener Remote Technologien ohne Programmierkenntnisse ermöglicht. Hier stieß ich zunächst auf konkrete Problematiken im Grenzbereich zwischen vollständig automatisierten Integrationsmechanismen, welche zunächst die Komplexität entsprechender Services lediglich auf die Ebene des Komponentenmodells und speziell die Parameter eines Dienstes abbildeten, was hinsichtlich der Nutzung keinen nennenswerten Fortschritt darstellte. Eine daraus resultierende Abschätzung des Detailgrades hinsichtlich Benutzerfreundlichkeit auf der einen und Flexibilität auf der anderen Seite führte in Kombination mit verschiedenen Verfahren zur Prozesssynthese (vgl. Teil-

bereich 1) jedoch zu hervorragenden Ergebnissen. So konnte ich auf Basis von Wissensrepräsentationen in Form von Datenflussinformationen, Typ- und Aktionstaxonomien sowie einer erweiterbaren Service Bibliothek mit Hilfe von Wrapper-Komponenten die auftretende Dienstkomplexität bei der automatisierten Integration von externen Remote Services auf ein Minimum reduzieren. Eine detaillierte Beschreibung dieser Vorgehensweise findet sich in Abschnitt 5.2.2

3. **Praktische Evaluation:** Auf Basis empirischer Fallstudien (siehe Abschnitt 6) konnte ich die im vorherigen Abschnitt definierten Anforderungen belegen und z.T. modifizieren, bzw. speziellen Szenarien gegenüber anpassen. Dies wirkte sich direkt auf den Entwicklungsprozess der Remote Umgebung aus. Der Denk- und Schaffungsprozess des erarbeiteten Frameworks wurde maßgeblich durch den praktischen Einsatz erster Prototypen geprägt und transparent gemacht und untermauert den wissenschaftlichen Anspruch der Arbeit. Der Einsatz semantischer Technologie zur automatisierten Integration von Service Komponenten und deren synthetisierter Modellkomposition (vgl. Arbeitsschwerpunkt 2) wurde beispielsweise überwiegend durch die Teilnahme an der Semantic Web Services Challenge (SWSC)[2] motiviert und vorangetrieben. Hier bestanden vor allem Probleme in der Anforderung an eine nach Möglichkeit rein deklarative Systembeschreibung zur automatischen Mediation von Daten- und Prozessflüssen, welcher die Remote Service Umgebung durch Einsatz o.a. Technologien größtenteils gerecht werden konnte. Zudem konnte ich die Praxistauglichkeit und industrielle Relevanz des entwickelten Gesamtsystems anhand der komplexen Szenarien der SWSC und anderen Fallstudien demonstrieren. Mit Hilfe grundverschiedener Domänen veranschaulichte ich u.a. die Flexibilität und die unterdessen eingetretene Akzeptanz der entstandenen Umgebung innerhalb unterschiedlicher Communities. Neben der wissenschaftlichen Anwendung meiner Arbeit belegen die Fallstudien zudem wesentliche Teilaspekte, die zu spezifischen Entscheidungen hinsichtlich der Architektur und Ausarbeitung der Remote Service Umgebung geführt haben. Vorgestellt werden Arbeiten zur automatischen Daten- und Prozessmediation sowie Service Discovery im Kontext der Semantic Web Services Challenge (Abschnitt 6.3), Fallstudien zu leichtgewichtigen Web Mashups (Abschnitt 6.5), statistische Auswertungen und der Einsatz von Web Service Bibliotheken im Rahmen biologischer Forschungsarbeiten (Abschnitt 6.1) und die Anwendung formaler Methoden (insbesondere Model Checking) innerhalb der FMICS Community (Abschnitt 6.2). Im Kontext der unterschiedlichen Domänen übernahm ich jeweils abwechselnd die Rolle des Domänen- oder Applikationsexperten sowie die des Service Anbieters. Hier galt es neben der Einarbeitung in die jeweiligen Domänen zunächst die formalen Problemspezifikationen

[2] `http://sws-challenge.org`

zu erarbeiten, adäquate Ontologien und Prozessmodelle zu entwerfen und die Methoden und Verfahren aus den Arbeitsschwerpunkten 1 und 2 anzuwenden und umzusetzen, was mir die auftretenden Probleme während der Analyse-, Modellierungs- und Implementierungsphase aus unterschiedlichen Blickwinkeln offenbarte. Hier wurde anhand von Praxiserfahrungen vor allem deutlich, dass nicht allein das entsprechende Werkzeug für einen serviceorientierten Lösungsansatz relevant ist, sondern insbesondere die angewandte Methodik und Denkweise der unterschiedlichen Nutzergruppen (durchaus softwaregestützt) optimiert werden muss. Daraus folgend leitete sich für mich der Anspruch ab, die Remote Service Umgebung mit konsequenten Hilfestellungen und sinnvollen Abstraktionen/Restriktionen dem Benutzer gegenüber als eine methodische Führungshilfe zu gestalten.

Im weiteren Verlauf dieser Arbeit soll nun in Kapitel 2 zunächst ein einleitender Überblick über verschiedene Technologien und Frameworks zur Umsetzung von MDSD und ihre Fähigkeit zur Integration verteilter Service Komponenten gegeben werden. Kapitel 3 befasst sich anhand ausgewählter Beispiele mit gängigen Technologien zur verteilten Bereitstellung, Ausführung und Anwendung von Service Komponenten nach deren Erläuterung in Abschnitt 5 der grundlegende Aufbau der Remote Service Umgebung beschrieben wird. Es folgen detaillierte Darstellungen durchgeführter Fachprojekte und Fallstudien (Abschnitt 6), anhand derer der Ausbau der Remote Service Umgebung begründet wird und Einzelkonzepte detailliert beschrieben werden. Insbesondere ermöglicht dieser Abschnitt darüber hinaus eine Einordnung in den wissenschaftlichen Kontext und mündet in einem Vergleich mit anderen gängigen Technologien und Entwicklungsumgebungen (Abschnitt 7). Abschließend wird in Abschnitt 8 ein Fazit gezogen und die erarbeiteten Ergebnisse analysiert, evaluiert und reflektiert.
Insbesondere die Arbeit an den Fallstudien, die kontinuierliche Weiterentwicklung der Remote Service Umgebung sowie die Einordnung in den wissenschaftlichen Kontext werden dabei durch eine Reihe meiner bisher publizierten Veröffentlichungen abgedeckt und gestützt, welche als Teil dieser kumulativ verfassten Dissertation angesehen werden können und sollen. Dabei handelt es sich um die Veröffentlichungen [KMFS06, MKSN06, KJMS09, MKS08, KMSN08, MBK+08, BCV+08, KMS+08, KVW+08, KMK+08, MMK+09].

2. Werkzeuge zur modellgetriebenen Softwareentwicklung

Um den grundlegenden Anforderungen an eine intuitive und ganzheitliche Umgebung zur modellgetriebenen Entwicklung von verteilten Applikationen gerecht zu werden ist es grundsätzlich notwendig zunächst eine konzeptuelle Grundlage für die generelle Spezifikation von Prozessen auf Basis möglichst grafischer Modelle zu erarbeiten. Diesbezüglich sollen in diesem Kapitel zunächst grundlegende Technologien und Werkzeuge zur universellen und grafisch unterstützten Prozessdefinition evaluiert werden, um abschließend die technische Grundvoraussetzung für die Integration verteilter Servicekomponenten zu schaffen. Da es sich bei der zu entwickelnden Remote Service Umgebung generell um einen ganzheitlichen und universellen Ansatz zur modellgetriebenen Entwicklung und im besonderen hinsichtlich Anforderung T5 (siehe Abschnitt 1) um ein Framework zur Orchestrierung konkreter Service Komponenten handeln soll, werden ausschließlich domänenübergreifende Technologien verglichen, mit denen es direkt möglich ist, beliebige ausführbare Softwarekomponenten zu entwickeln. Fachspezifische und eher abstrakte Modellspezifikationen wie z.B. UML [con11], OSGI [Fun09] oder EPK [KNS92] finden daher keine Betrachtung. Ausgehend von dem in [Nag09] durchgeführten Vergleich verschiedener, grundlegender technologischer Ansätze zur grafischen Prozessdefinition und insbesondere mit Hinblick auf die Integration verteilter Service Komponenten soll an dieser Stelle der Fokus auf konkreten grafischen Modellierungswerkzeugen und Ihrer Handhabe in Bezug auf die Anforderungen aus Abschnitt 1 liegen.

Business Process Execution Language

Diskutiert man die Möglichkeiten in Bezug auf die Nutzung externer Dienste innerhalb von serviceorientierten Prozessdefinitionen, führt kein Weg an der Business Process Execution Language (BPEL) vorbei. BPEL gilt als Standard, wenn es um die Orchestrierung von Web Services geht, was ihren Ursprung in der Tatsache begründet sieht, dass die Sprache in gemeinschaftlicher Arbeit von BEA,

IBM und Microsoft als XML [(W312] Dialekt auf diese Anforderung hin, nämlich die Orchestrierung von Web Services, entwickelt und zugeschnitten wurde. Als zu vergleichende Basistechnologie nimmt BPEL hier jedoch eine Sonderstellung ein und soll an dieser Stelle nicht in Betracht gezogen werden, da es sich lediglich um die zu Grunde liegende Repräsentation von Prozessen innerhalb verschiedener grafischer Modellierungswerkzeuge (z.B. Netbeans Enterprise Pack[1] oder OMII-UK[2]) handelt und in erster Linie auf die Nutzung von reinen Web Services beschränkt ist. Zusätzliche Funktionalitäten hinsichtlich der Anforderungen aus Abschnitt 1, beispielsweise zum Ausführen (T7), Versionieren (A4) oder Beobachten (A3) von Prozessen, werden generell durch die konkreten Modellierungsumgebungen und/oder Laufzeitumgebungen bereitgestellt und haben nur indirekt mit der eingesetzten Prozessrepräsentation zu tun. Aus diesem Grund soll BPEL hier als assoziierter Standard im Kontext von verteilten Diensten zwar Erwähnung finden, jedoch nicht weitergehend verglichen werden. Eine Einordnung der Sprache in Bezug zur grafischen Prozessmodellierung im allgemeinen, und im Kontext der im weiteren Verlauf dieses Abschnitts ebenfalls angesprochenen jABC Modellierungsumgebung im besonderen, findet sich darüber hinaus in [Nag09]. Die im folgenden verglichene Anzahl von Modellierungswerkzeugen kann nur einen kleinen Ausschnitt der gegenwärtig verfügbaren Technologie repräsentieren und erhebt in keinster Weise den Anspruch auf Vollständigkeit. Der Fokus liegt auf frei verfügbaren Tools oder kommerziellen Anwendungen, bei denen der Hersteller zumindest die Möglichkeit zur Evaluation der Software einräumt.

2.1. jABC

Die jABC Modellierungsumgebung, entwickelt am Lehrstuhl für Programmiersysteme der Technischen Universität Dortmund[3] ist ein offenes Framework, welches aufgrund seiner auf der Java Technologie basierenden Codebasis vollständig plattformunabhängig ist. Dienste basieren auf gekapselten Java Klassen, was einen hohen Freiheitsgrad der Implementierung von Funktionalitäten gewährleistet. Eine Sammlung domänenunabhängiger und grundlegender Service Komponenten ist darüber hinaus bereits enthalten. Prozesse werden durch Kontrollflussgraphen als Kripke Transitionssystem (KTS) [MOSS99, MS09] repräsentiert und in einem proprietären XML Format hinterlegt. Dabei ist es bereits möglich, über ein Versionierungssystem verschiedene Entwicklungsstadien oder Abwandlungen eines Modells zu verwalten. Zusatzfunktionen gestatten überdies die Annotation von Kontrollflussinformationen, sollten diese benötigt werden. Die Applikation ist modular gestaltet und bietet über eine Plugin Schnittstelle die Möglichkeit zur

[1] `http://netbeans.org`
[2] `http://www.omii.ac.uk`
[3] `http://ls5-www.cs.tu-dortmund.de/opencms/de/ls5/`

nahezu beliebigen Erweiterung des Gesamtsystems. Eine Vielzahl solcher existierender Zusatzkomponenten, wie die o.a. Funktion zur Behandlung von Kontrollflussinformationen, ermöglicht überdies bereits die Generierung diverser Prozessausgangsformate, die formale Verifikation von lokalen Diensten und globalen Modellen (Local Checking, Model Checking) sowie die Integration einer proprietären Lösung zur Ausführung verteilter Service Komponenten. Zudem ist eine integrierte wie auch extern verwendbare Laufzeitumgebung für ausführbare Arbeitsabläufe vorhanden. Die Plugins haben zusätzlich Einfluss auf die konkrete Semantik eines Modells. So kann ein Prozess beispielsweise je nach Einsatzbereich und Domänenspezifikation als Zustandsautomat, Syntaxbaum, Applikation, o.ä. angesehen werden. Die jeweilige Interpretation des Modells obliegt dabei dem entsprechenden Plugin.

2.2. MetaEdit+

MetaEdit+[4], entwickelt von Metacase, gliedert sich grob in zwei Teilbereiche: die Workbench und den Modeler. Erste ist im Sinne des domänenspezifischen Modellierens (DSM) für die Erstellung der Modellierungssprache (DSL) verantwortlich. Das resultierende Metamodell fungiert anschließend als Eingabe für den Modeler, welcher die eigentliche Umgebung zur grafischen Orchestrierung von Diensten darstellt. Aufgrund der frei definierbaren Metamodelle ist MetaEdit+ in der Lage beliebige Prozessrepräsentationen zu verarbeiten. Arbeitsabläufe können im allgemeinen datenflussspezifisch oder als Kontrollflussstruktur und daraus resultierend im besonderen auch in anerkannten Standardformaten wie z.B. der Business Process Modeling Notation (BPMN) modelliert werden. Die Modelle werden generell innerhalb eines proprietären XML-Formats persistiert und können über Code-Generatoren in verschiedene Ausgangsformate (z.B. BPEL) umgewandelt werden. Dienste werden durch frei programmierbare Service Komponenten repräsentiert. Eine Möglichkeit zur Anbindung externer Dienstleistungen ohne die manuelle Schnittstellenimplementierung innerhalb der Services ist jedoch nicht möglich. Ebenso gibt es keine Funktionalität zur Verifikation globaler Spezifikationen über das gesamte Modell. Eine einfache Erweiterung des Systems auf Basis einer offenen API ist nicht vorgesehen, was die Anbindung von zusätzlichen Funktionen erschwert. MetaEdit+ ist auf allen gängigen Plattformen (Windows, Linux, Mac OS X) verfügbar, jedoch nicht vollständig plattformunabhängig.

[4] `http://www.metacase.com/`

2.3. Taverna Workflow System

Das Taverna Workflow System [OAF+04], bestehend aus der Taverna Workbench und Taverna Server, ist neben Kepler[5], Triana[6] und Wildfire[7] ein weit verbreitetes Modellierungswerkzeug zur Orchestrierung von Diensten im Kontext biotechnologischer Arbeitsabläufe. Ohne auf die Domäne der Bioinformatik beschränkt zu sein liegt der Fokus der Applikation ganz klar auf diesem Fachgebiet. Dies wird insbesondere durch die Auswahl mitgelieferter Service Komponenten und angebundener Verzeichnisdienste für externe Dienste deutlich, die ausschließlich Funktionalitäten aus dem Bereich der Biotechnologie bereitstellen. Taverna Workbench beinhaltet bereits Möglichkeiten zur Anbindung verteilter Dienste in Form von Web Services sowie eine externe Laufzeitumgebung (Taverna Server) zur Bereitstellung und Ausführung modellierter Arbeitsabläufe. Einzelne Dienste werden als XML Fragmente, sog. Prozessoren, repräsentiert innerhalb derer die konkreten Funktionsaufrufe stattfinden. Komplexe Prozesse werden innerhalb eines proprietären XML Formats hinterlegt. Die Modellierung erfolgt dabei datenflussorientiert mit Hilfe der aus dem myGrid Projekt übernommenen Simple Conceptual Unified Flow Language (SCUFL), bzw. deren Nachfolger SCUFL2[8]. Hinsichtlich der Manipulation des Kontrollflusses einer Applikation bietet Taverna lediglich eingeschränkte Möglichkeiten. So können beispielsweise manuell erzeugte Ausführungsskripte in einen Arbeitsablauf eingefügt oder spezielle Prozessoren als Workaround eingesetzt werden, was jedoch zu einem Aufweichen des MDSD Konzeptes führt und die Nutzung für Anwendungs- und Domänenexperten ohne Programmierkenntnisse zusätzlich erschwert. Diese eher technische Art der Modellierung trifft auf alle ähnlichen und zu Beginn genannten Modellierungswerkzeuge aus dem Bereich der Bioinformatik zu, weshalb Taverna an dieser Stelle beispielhaft für diese Gruppe stehen soll. Das System beinhaltet rudimentäre Funktionalitäten zur Syntaxverifikation der Servicedefinitionen aber überdies keinerlei Möglichkeiten zur Überprüfung von Spezifikationen auf dem gesamten Modell. Taverna ist, wie auch MetaEdit+, für alle gängigen Plattformen (Windows, Linux, Mac OS X) verfügbar, jedoch nicht vollständig plattformunabhängig. Erweiterungen an Taverna werden über ein Pluginkonzept realisiert, was einen einfachen Ausbau der Kernfunktionalitäten gestattet.

[5] `https://kepler-project.org/`
[6] `http://www.trianacode.org/`
[7] `http://wildfire.bii.a-star.edu.sg/`
[8] `http://www.gridworkflow.org/`

2.4. Windows Workflow Foundation

Entwickelt von Microsoft und als Bestandteil der .NET Umgebung[9] stellt sich die Windows Workflow Foundation[10] als Werkzeug zur grafischen Modellierung und Ausführung von Arbeitsabläufen dar. Als Dienste können dabei beliebig implementierte .NET Komponenten eingesetzt werden. Das System beinhaltet ebenfalls Funktionalitäten zur einfachen Anbindung verteilter Dienste, ist aber auf .NET Dienste und Microsofts Windows Betriebssystem beschränkt, was eine modulare Anbindung verschiedener Service Komponenten als auch eine plattformunabhängige Verbreitung der Modellierungsumgebung erschwert, bzw. verhindert. Prozesse werden entweder als endliche Automaten mit Ereignissen, persistenten Zuständen und Zustandsübergängen oder als Kontrollflussgraph modelliert.Als zugrunde liegende Struktur für die Prozessmodellierung dient XAML [Mic12], ein XML Format. Verifikationen von Modellspezifikationen sind nicht vorgesehen. Obwohl domänenunabhängig zielt die Windows Workflow Foundation im Wesentlichen auf die vereinfachte Automatisierung von Arbeitsabläufen zwischen Microsoft Applikationen (z.B. MS Office) ab. Eine externe Laufzeitumgebung zur Ausführung von Prozessen ist u.a. innerhalb des MS SharePoint[11] Server integriert. Weiterführende Funktionserweiterungen in Form von Plugins o.ä. sind nicht vorgesehen, was eine Erweiterung des Systems erschwert, bzw. gänzlich verhindert.

2.5. Drools Flow

Die Flow-Erweiterung der JBoss Drools Plattform[12] zur Verarbeitung von Regeln, Events und Arbeitsabläufen bietet Möglichkeiten zur grafischen Modellierung von (Business-)Prozessen als Flussgraphen. Drools Flow versteht sich dabei nicht als ganzheitlicher Ansatz zum Entwurf komplexer Applikationen, sondern vielmehr als Framework zur Konstruktion und Integration separierter Business-Logik Teile in bestehende Applikationsszenarien. Dabei bietet die Umgebung die Möglichkeit zur grafischen Spezifikation mit Hilfe moderner UI Konzepte wie z.B. Drag & Drop, zur grafischen Ausführung mittels eines Debuggers und Business Activity Monitoring zur Überwachung der jeweiligen Prozesse. Drools Flow nutzt grafische Bausteine, sog. Building Blocks, zur grafischen Modellierung der Prozesse. Die Palette an vorgefertigten Bausteinen umfasst dabei Standardkomponenten wie z.B. Schleifen, Fork-Join Elemente oder auch Komponenten zur hierarchischen Abbildung von Sub-Modellen. Als Community Projekt legt das Flow-Projekt zu-

[9] `http://msdn.microsoft.com/de-de/library/bb979310.aspx`
[10] `http://msdn.microsoft.com/de-de/library/dd489441.aspx`
[11] `http://sharepoint.microsoft.com/`
[12] `http://www.jboss.org/drools/drools-flow`

dem Wert auf eine individuelle Erweiterung dieser Standardbibliothek. So werden insbesondere DSL in Form von Building Blocks integriert. Als Bestandteil der Drools Umgebung versteht sich das Framework zudem auf die Auswertung von Business Regeln im Sinne eines Entscheidungssystems. Die Prozessmodellierung wird dabei lose mit regelbasierten Auswertungsmechanismen gekoppelt und bietet somit Möglichkeiten zur Entscheidungsfindung innerhalb der Ausführung von Prozessen. Beispielsweise können Subprozesse oder menschliche Interaktion automatisiert auf Basis von spezifizierten Regeln delegiert werden.
Um die proprietäre Modellrepräsentation an bestehende Industriestandards wie z.B. BPEL [con03], bzw. BPMN [gro11a] anzupassen werden innerhalb des Frameworks verschiedene Skins eingesetzt, um die eigenen Konzepte auf äquivalente Visualisierungen in der Zielnotation abzubilden. Der zunächst eingeschlagene Weg, Drools Flow mit Hilfe eines Skin die Modellierung von BPMN Prozessen zu ermöglichen wurde mittlerweile verworfen, da Drools Flow unterdessen gänzlich in dem nebenläufigen Schwesterprojekt jBPM[13] aufgegangen ist und nicht eigenständig weiterentwickelt wird.

2.6. Yahoo Pipes

Das Yahoo Pipes Modellierungswerkzeug[14] ist als reine Web-Applikation konzipiert und ausschließlich online verfügbar. Es zielt in erster Linie darauf ab, populäre Web Anwendungen auf Basis ihres Datenflusses zu sog. Mashups miteinander zu verbinden und komplexe individuelle Funktionalitäten daraus abzuleiten. Das freie Definieren von beliebigen Diensten ist nicht vorgesehen. Lediglich eine Auswahl an Standardkomponenten zur Dateneingabe/-ausgabe, zum Auslesen von RSS Feeds oder zur Ansteuerung von Anwendungen wie Google Search oder Flickr werden mitgeliefert. Erzeugte Mashups können gespeichert und anderen Nutzern der Umgebung als atomarer Dienst zur Verfügung gestellt werden. Weiterführende Möglichkeiten zur Verifikation von Modellen, zum Export verschiedener Ausgangsformate oder Erweitern der Modellierungsumgebung sind nicht vorgesehen. Alles in allem soll Yahoo Pipes nicht als allgemeines MDSD Modellierungswerkzeug gelten, sondern vielmehr eine einfache Möglichkeit zur Kombination aktueller, meist aus dem Umfeld sozialer Netzwerke stammender, Web Anwendungen dienen. Innerhalb dieser Domäne führt der auf ein Minimum reduzierte Funktionsumfang allerdings zu einer sehr einfachen und leicht zu erlernenden Bedienbarkeit.

[13] `http://www.jboss.org/jbpm`
[14] `http://pipes.yahoo.com/`

2.7. Auswertung

Neben der generellen Handhabung und Erweiterbarkeit der jeweiligen Modellierungswerkzeuge, resultieren die jeweiligen Eigenschaften hinsichtlich Nutzung und Funktionsumfang in entsprechend mehr oder weniger ausgeprägten Vorbedingungen hinsichtlich der in Abschnitt 1 definierten grundlegenden Anforderungen.
Yahoo Pipes bietet aufgrund seines beschränkten Dienstverzeichnisses, der vollständigen Abhängigkeit der Verfügbarkeit von Drittanbietern zur Laufzeit sowie mangels der Möglichkeit, beliebige Service Komponenten zu integrieren keinerlei Modularität, Kompositionalität (A1) oder Universalität (A2). Eine Möglichkeit zur Überwachung von Ausführungsabläufen oder vergleichende Leistungstests (A3) fehlen ebenso wie die versionsorientierte Verwaltung von Modellen (A4). Lediglich in puncto Plattformunabhängigkeit (A5) ist durch das konsequente Implementieren als reine Web-Applikation ein hohes Maß an Flexibilität gewährleistet. Die Integration von zusätzlichen Komponenten über eine definierte Schnittstelle ist nicht möglich (A6).
Basierend auf Java Technologie und offenen Standards sowie durch die modulare Kapselung der Service Komponenten in Building Blocks erfüllt **Drools Flow** zwar die Anforderungen an Modularität (A1), Universalität (A2) und die Plattformunabhängigkeit (A5), mangels eines Mechanismus zur Verifikation von Diensteigenschaften, bzw. der expliziten, semantischen Bedeutung der verwendeten Regeln jedoch nicht diejenigen an die Kompositionalität (A1). Mit Hilfe des integrierten Business Activity Monitoring ist zudem eine Überwachung der Prozesse möglich (A3). Der Abruf von History Informationen und eine Versionierungen der Prozesse sind ebenfalls standardmässig durchführbar (A4). Als ursprüngliches Community-Projekt und im Rahmen der JBoss Entwicklung weitergeführtes Werkzeug, bietet die Modellierungsumgebung eine gute Erweiterbarkeit. Eigene Building Blocks sind einfach zu integrieren und die Schnittstellen der Plattform sind gut dokumentiert und zugänglich.
Die **Microsoft Windows Workflow Foundation** genügt aufgrund Ihrer Flexibilität im Rahmen des .NET Frameworks den Anforderungen an eine modulare, kompositionelle (A1) und universelle (A2) Modellierungsumgebung. Durch die Einschränkung auf .NET relativiert sich dieser Vorteil allerdings im direkten Vergleich mit anderen Werkzeugen, welche an dieser Stelle ein deutlich offeneres Konzept bieten. Mit den über die Visual Studio Suite[15] bereitgestellten Zusatzkomponenten ist ein Debugging und Überwachen von Prozessausführungen (A3) möglich. Eine Möglichkeit zum direkten Leistungsvergleich fehlt allerdings auch hier. Die Versionierung der Modelle (A4) ist mit den mitgelieferten Werkzeugen der Windows Workflow Foundation nicht möglich. In puncto Plattformunabhängigkeit (A5) und Erweiterbarkeit (A6) ist die Modellierungsumgebung als geschlossenes und auf die Microsoft Windows Umgebung beschränktes System zu

[15] `http://www.microsoft.com/germany/visualstudio/`

verstehen.
Taverna besitzt aufgrund des Konzepts von als sog. Prozessoren gekapselten Diensten gewisse Einschränkungen hinsichtlich der Modularität/Kompositionalität (A1) und Universalität (A2). Generell kann direkt kein funktionaler Programmcode innerhalb der Service Komponenten aufgerufen werden. Eine Erweiterung zur strukturierten Überwachung von Prozessausführungen (A3) oder Verwaltung mehrerer Versionen eines Modells (A4) ist nicht unmittelbar vorhanden. Die Plattformunabhängigkeit (A5) des Systems beschränkt sich auf die drei gängigen Betriebssysteme. In puncto Erweiterbarkeit kann das offene Pluginkonzept überzeugen.
MetaEdit+ besitzt hinsichtlich seiner Flexibilität in Bezug auf frei definierbare Dienste und beliebige Metamodelle ein hohes Maß an Modularität, Kompositionalität (A1) und Universalität (A2). Eine unmittelbare Funktion zum Monitoring oder Benchmarking von Prozessen (A3) existiert nicht. Ebenso fehlt die Möglichkeit zur versionierten Verwaltung von Modellen (A4). Die Plattformunabhängigkeit (A5) beschränkt sich auf die drei verbreitetsten Betriebssysteme Linux, Mac OSX und Microsoft Windows. Aufgrund des kommerziellen Charakters der Software ist die Erweiterbarkeit (A6) stark eingeschränkt.
Da die **jABC** Modellierungsumgebung in ihren grundlegenden Anforderungen weitestgehend mit denen in dieser Arbeit geforderten übereinstimmt und hinsichtlich deren Erfüllung entwickelt wurde, schneidet sie im Vergleich zu den anderen betrachteten Frameworks überdurchschnittlich gut ab. Das völlig freie Definieren von Diensten auf Basis von Java Klassen sowie das konsequente Umsetzen der losen Kopplung von Service Komponenten gewährleistet ein Maximum an Modularität, Kompositionalität (A1) und Universalität (A2). Eine integrierte Ausführungskomponente mit Funktionen zur detaillierten Überwachung von Prozessausführungen (A3) ist ebenso vorhanden wie eine komplette Versionierung der modellierten Arbeitsabläufe (A4). Durch den durchgängigen Einsatz der Java Technologie wird eine größtmögliche Plattformunabhängigkeit erreicht (A5) und das offene und modulare Pluginkonzept lässt Raum für individuelle Funktionserweiterungen des Systems (A6). Eine Vielzahl solcher bereits verfügbarer Plugins decken zudem weitere Anforderungen insbesondere hinsichtlich der Verifikation von Modellen (A7) und Diensten (vgl. A1) mit Hilfe von Model Checking Mechanismen ab. Lediglich in puncto Benchmarking von verschiedenen Prozessausführungen existiert zum jetzigen Zeitpunkt keine adäquate Komponente innerhalb der Modellierungsumgebung.
Die Ergebnisse des Vergleichs sind in Tabelle 2.1 zusammengefasst. Neben dem jABC, welches allen Kriterien zumindest teilweise genügt (A3), erfüllt einzig Drools Flow, bzw. dessen Nachfolgeprojekt jBPM einen Großteil der Anforderungen. Unabhängig davon ist der größte Nachteil dieses Frameworks jedoch, dass kein ganzheitlicher Ansatz, sondern die Erweiterung und Anreicherung bestehender Applikationen im Vordergrund steht. Aus diesem Grund und dem Umstand, dass im unmittelbaren Umfeld meiner Arbeitsgruppe das entwickelt wird und

Anforderung	jABC	MetaEdit+	Taverna	WWF	Drools	Pipes
A1 (Modul./Komp.)	●	●	◉	◉	◉	○
A2 (Universalität)	●	●	◉	◉	●	○
A3 (Monitoring)	◉	◉	○	◉	●	○
A4 (Versionierung)	●	○	○	○	●	○
A5 (Plattformunabh.)	●	◉	◉	○	●	●
A6 (Erweiterbarkeit)	●	○	●	○	●	○
A7 (Verifikation)	●	○	○	○	○	○

Tabelle 2.1.: Vergleich relevanter Modellierungswerkzeuge hinsichtliche der grundlegenden Anforderungen aus Abschnitt 1

● erfüllt
◉ teilweise erfüllt
○ nicht erfüllt

daher uneingeschränkter Zugang zu Technologie und Wissensaustausch besteht, soll an dieser Stelle die jABC Modellierungsumgebung favorisiert werden. Zudem steht neben der Tatsache, dass mit dem System ebenfalls bereits ein Großteil der grundlegenden Anforderungen an eine verteilte Remote Service Umgebung abgedeckt werden (bis auf die leichten Defizite des Monitoring während der Ausführung von Prozessen), der Umstand fest, dass bereits ein Basiskonzept zur Erweiterung des jABC um ein Remote Framework existiert. Historisch betrachtet existiert eine proprietäre Implementierung einer Umgebung zur Integration verteilter Dienstleistungen für das ABC, das Vorgängerprojekt des jABC. Dieses Framework mit der Bezeichnung ETI soll im weiteren Verlauf der Arbeit in Kombination mit dem jABC als konzeptuelle Grundlage dienen und wird im Zuge der detaillierten Beschreibung des hier entwickelten Systems (per Namenskonvention nun jETI genannt) in Abschnitt 5 vorgestellt.

3. Remote Service Technologien

Zu Beginn der Konzeption einer ganzheitlichen Umgebung zur Integration von Remote Services steht neben der in Abschnitt 2 durchgeführten Evaluation einer Basisplattform eine Bestandsaufnahme aktueller und etablierter Standardtechnologien zur Ausführung verteilter Softwarekomponenten. Da das zu erarbeitende Framework im Wesentlichen modular aufgebaut und generell offen für neue technologische Ansätze gestaltet werden soll, wird im folgenden allerdings lediglich eine Auswahl gängiger Konzepte zum verteilten Bereitstellen und Ausführen von Funktionalitäten aufgezeigt, welche durch adäquate Beispiele veranschaulicht werden. Ausgehend vom aktuellen Stand der Technologie existiert historisch bedingt eine ganze Reihe verschiedenster Möglichkeiten, um Software Komponenten verteilt miteinander zu verknüpfen und entfernte Funktionalitäten zugreifbar zu machen. Diese sind in den meisten Fällen auf spezielle Anforderungen einer bestimmten Domäne zugeschnitten und divergieren in der Art ihrer Nutzung als auch der angebotenen technischen Möglichkeiten.

3.1. Remote Procedure Call

Als *Remote Procedure Call* (RPC) [rpc76, rpc88, Sri95b] bezeichnet man im allgemeinen eine spezielle Technik der Interprozesskommunikation, bei der es um den Aufruf entfernt verfügbarer Funktionen oder Programmroutinen geht. Übertragen auf den serviceorientierten Ansatz entspricht dieses Vorgehen prinzipiell dem Aufruf eines entsprechend parametrisierten Dienstes. RPC hat sich innerhalb der letzten drei Jahrzehnte aufgrund seines zuverlässigen und gut dokumentierten Verfahrens als ein Standard zur Kommunikation verteilter Softwarekomponenten etabliert. Zudem basieren moderne und populäre Remote Technologien in weiten Teilen auf diesem Konzept und sind z.T. durch die direkte Integration in Programmiersprachen sehr einfach und effizient einzusetzen.
Grundsätzlich handelt es sich bei RPC um ein Client-Server-Modell, bei dem der Client (Dienstnutzer) eine Anfrage an einen zuvor bekannten Server (Dienstanbieter) schickt und die entsprechende Antwort abwartet. Die Anfrage setzt sich dabei aus der eindeutigen Signatur der aufzurufenden Funktion zusammen, also der Name oder die Identifikationsnummer einer auszuführenden Routine, sowie deren Parameter. Um sicherzustellen, dass ein Server genau die angefragte Funk-

tionalität bereitstellt, macht dieser sein Dienstangebot zuvor bei einem Verzeichnisdienst bekannt, welcher vom Client vor dem Absenden der Anfrage konsultiert wird. Alternativ kann in einer lokal vernetzten Umgebung auch ein direkter Rundruf des Clients an alle verfügbaren Server gestartet werden. Nachdem ein Server die gewünschte Operation entsprechend der Anfrage bearbeitet hat, schickt er das Ergebnis als Antwort zurück zum Client. Das Konzept erlaubt es dabei komplexe Daten unterschiedlichen Typs auszutauschen. Aufrufe können nach dem „call-by-value“ oder „call-by-reference“ Prinzip erfolgen, bei dem entweder Kopien der übermittelten Werte, bzw. gänzlich neue Datentypen übermittelt werden oder aber die entsprechenden Parameter direkt referenziert verändert werden. Aufgrund problematischer, bzw. ineffizienter Realisierungen des call-by-reference Prinzips, schränken viele RPC Implementierungen die Nutzung auf call-by-value ein.
RPC wird aufgrund seiner Struktur generell in der klassischen Programmierung (siehe 3.1.1) zum internen Aufruf von Methoden und Austausch von Daten/Objekten unterschiedlichen Typs über ein Netzwerk verwendet. Die Nutzung sollte sich dabei idealerweise analog zum Aufruf einer Prozedur innerhalb der verwendeten Programmiersprache (je nach Implementierung des RPC) richten, woher sich auch der Name des entfernten Prozeduraufrufs herleitet. Der Datenaustausch eines RPC kann auf unterschiedlichen Kommunikationsprotokollen beruhen (z.B. TCP oder UDP) und verschiedene Sicherheitsmechanismen beinhalten (Verschlüsselung, Authentifizierung). Treten bei einem Aufruf keine Fehler auf, garantiert ein RPC immer eine einmalige Ausführung der angefragten Operation. Auf der Basis verschiedener RPC-Fehlersemantiken (Maybe, At-Least-Once, At-Most-Once, Exactly-Once) [rpc04], welche das strukturelle Verhalten eines kompletten Aufrufs beschreiben, ist zudem eine Problembehandlung vorgesehen. Im Fehlerfall verhalten sich die Clients verschiedener RPC Varianten dabei entsprechend der in [rpc04] genannten Prinzipien, was hier nicht weiter ausgeführt werden soll.

3.1.1. Einsatz der Technologie

Seit der grundlegenden Formulierung der Idee in den 1970er Jahren haben sich verschiedene RPC-Implementierungen etabliert. Dazu zählt z.B. das Open Network Computing Remote Procedure Call Verfahren (ONC RPC) [rpc88, Sri95b, Cer95, Sri95a, Chi99], etwickelt von SUN Microsystems und daher auch als SUN RPC bekannt, oder die Distributed Computing Environment Remote Procedure Call Implementierung (DCE RPC) [Gro98], welche von Microsoft vorgelegt wurde. Letzteres bieten dabei in weiten Teilen die Grundlage des Microsoft DCOM Standards, bzw. eine der Remote Funktionalitäten der .NET Umgebung zur Interprozesskommunikation in Microsoft Windows Umgebungen. Weitere relevante und weit verbreitete Implementierungen des RPC Konzepts sind die *Java Remote*

Method Invocation (RMI) zur Kommunikation zwischen Java Objekten, welche im folgenden in Abschnitt 3.1.2 vorgestellt wird, und die *Common Object Request Broker Architecture* (CORBA) [gro11b, gro11c, gro11d] zur plattformunabhängigen Kommunikation zwischen Objekten verschiedener objektorientierter Sprachen. Andere Derivate wie RPyC[1] zur Kommunikation innerhalb der Python[2] Programmiersprache oder XML-RPC [Win99] als RPC System auf Basis von XML und HTTP [BLFF96] sollen der Vollständigkeit halber Erwähnung finden, spielen aber an dieser Stelle keine Rolle.

3.1.2. Beispielimplementierung: Java RMI

Mit der Remote Method Invocation (RMI) [Ora12] bietet Java eine Erweiterung des RPC Konzepts zur direkten und vollständig integrierten Kommunikation zwischen Objekten auf der Implementierungsebene. RPC bietet bereits die Möglichkeit entfernte Aufrufe der Art zu abstrahieren, dass der Nutzer keine Unterschiede zu einer lokalen Ausführungsroutine bemerkt. Dabei werden die Argumente und Rückgabewerte des Aufrufs durch den Client technisch entsprechend gekapselt und auf Basis einer externen Datenrepräsentation, wie z.B. der EXternal Data Representation (XDR) [xdr87, Eis06] übermittelt. Die Schwächen von RPC als ursprünglich nicht objektorientierte Technologie liegen allerdings konzeptuell und historisch bedingt in der direkten Kommunikation zwischen Objekten in verschiedenen Adressräumen auf der Ebene der Programmiersprache. Um das RPC Konzept auf die Semantik von Objektaufrufen zu übertragen wurde RMI von Sun Microsystem eingeführt und spezifisch auf die Java Programmiersprache ausgerichtet. Der hohe Integrationsgrad führt dabei allerdings zu einer homogenen Struktur der verteilten Komponenten, welche alle auf den Kontext virtueller Java Maschinen beschränkt sind, jedoch uneingeschränkten Zugriff auf das Java Objektmodell ermöglichen. Neben der generellen Anforderung eines einfach zu nutzenden aber natürlich integrierten Remote Konzepts werden die dedizierten Anforderungen an RMI wie folgt angegeben:

- Unterstützung nahtloser Integration von Objektaufrufen über die Grenzen verschiedener virtueller Maschinen hinaus
- Integration eines verteilten Objektmodells bei gleichzeitiger Beibehaltung der Semantik lokaler Objekte
- Transparente Abgrenzung zwischen verteiltem und lokalem Objektmodell
- Vereinfachung der Implementierung und Handhabung von verteilten Systemkomponenten

[1] `http://rpyc.sourceforge.net/`
[2] `http://python.org/`

- Beibehaltung der durch die Java Laufzeitumgebung bereitgestellten Typsicherheit
- Unterstützung verschiedener Arten von Objektreferenzen (nicht-persistent, persistent)
- Beibehaltung der gebotenen Sicherheitskonzepte durch die Java Umgebung in Form von Security Manager Komponenten und Classloader

3.1.2.1. Genereller Aufbau einer RMI Applikation

Aufbauend auf dem RPC Konzept bestehen RMI fähige Applikationen meist aus einer Anzahl von Client- und Serverkomponenten. Typischerweise wird dabei vom Server eine Reihe an Remote Objekten bereitgestellt, welche über verteilte Referenzierungsschnittstellen zugreifbar gemacht werden. Die Clients bedienen sich dieser Schnittstellen, um entsprechende Operationen auf den Objektinstanzen auszuführen. Die Kommunikation zwischen den Parteien wird durch RMI komplett gekapselt und vor dem Benutzer verdeckt. Daraus ergibt sich ein analog zu lokalen Objekten aufgebauten Nutzungsverhalten. Der Aufruf eines verteilten Objektes entspricht aus der Nutzersicht somit dem einer lokalen Objektinstanz. Das Auffinden der angebotenen Objekte geschieht über deren serverseitige Bekanntmachung innerhalb eines RMI eigenen Verzeichnisdienstes (RMI-Registry). Clients können über Anfragen an diesen Dienst entsprechende Objektreferenzen finden und nutzen. Die generelle Architektur einer RMI Applikation ist in Abb. 3.1 dargestellt. Konzeptuell ist der größte Vorteil von RMI gleichzeitig auch der gravierendste Nachteil. Die sehr konkrete Integration in die Java Programmiersprache vereinfacht den Umgang zwischen geeigneten Softwarekomponenten immens. Allerdings beschränkt die Technologie den Benutzer auf die Verwendung entsprechender Java Applikationen in einem homogenen Umfeld. Zudem können bestehende lokale Funktionalitäten nicht ohne komplexe Eingriffe in die Struktur eines Programms zur verteilten Verwendung nutzbar gemacht werden.
Als Basistechnologie für die Java zu Java Kommunikation der zu entwickelnden Remote Service Umgebung soll RMI aus Gründen der Performance und Usability Einzug halten. Durch die nahtlose und direkte Anwendung innerhalb der Implementierungsebene erzielt die Technologie neben Ihrer einfachen Art der Anwendung vor allem in puncto Ausführungszeit sehr gute Performance Werte im Vergleich zu anderen Remote Technologien wie z.B. Web Services. Moderne CORBA Implementierungen kommen zwar an diese Werte heran, gehen aber mit einem erhöhten Maß an Komplexität einher.

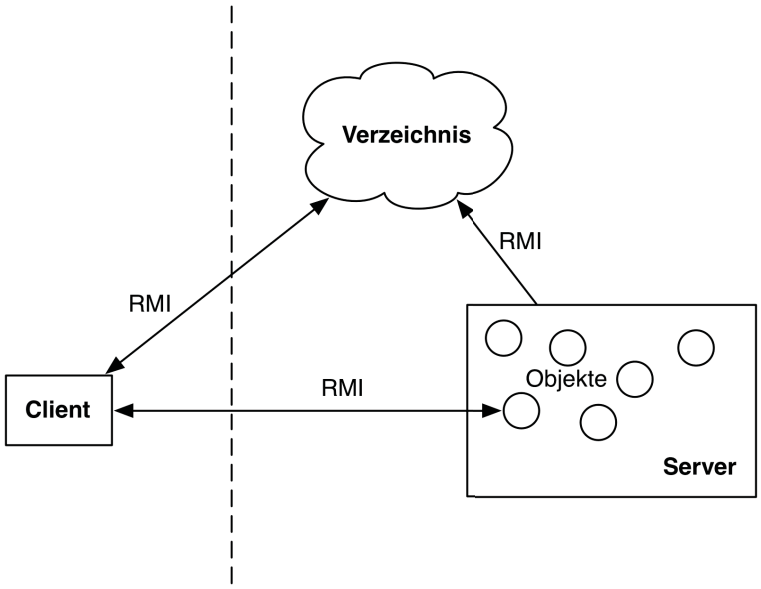

Abbildung 3.1.: Architektur einer RMI Applikation

3.2. Web Services

Web Services [(W311, Pap06] gelten heute weitestgehend als State of the Art, wenn es um die verteilte Anbindung externer und komplexer Funktionalitäten innerhalb einer serviceorientierten Umgebung geht. Innerhalb einer SOA besitzen sie eine hohe Relevanz und werden beispielsweise als Basistechnologie der Serviceimplementierungen eines BPEL Prozesses eingesetzt.
Als klassischer Web Service werden i.d.R. serverseitige Dienste angesehen, welche neben ihrer bereits durch die Namensgebung implizierten Eigenschaft, über ein Netzwerk, üblicherweise das World Wide Web, verfügbar zu sein, eine Schnittstellenbeschreibung in Form eines Web Service Description Language (WSDL) [CCMW01] Dokuments besitzen und über XML-RPC, bzw. SOAP [BEK+00] miteinander kommunizieren. Eine WSDL Spezifikation ist dabei eine strukturierte Definition der Eigenschaften, Funktionen und verwendeten Datentypen eines Dienstes, basiert syntaktisch auf XML und ermöglicht einen sehr hohen Freiheitsgrad bei der Spezifikation von Servicekomponenten. Zur Verfügung gestellte Dienste können über Verzeichnisdienste, wie z.B. Universal Description, Discovery and Integration (UDDI) [con02a] bekannt gemacht werden, was das clientseitige Auffinden entsprechender Funktionalitäten ermöglicht. Die komplette Architektur eines Web Service Szenarios ist in Abb. 3.2 dargestellt. Web Services werden durch Ihre jeweilige Schnittstellenbeschreibung hinsichtlich des Dienstaufrufs und der verwendeten Datentypen komplett strukturell beschrieben, was eine automati-

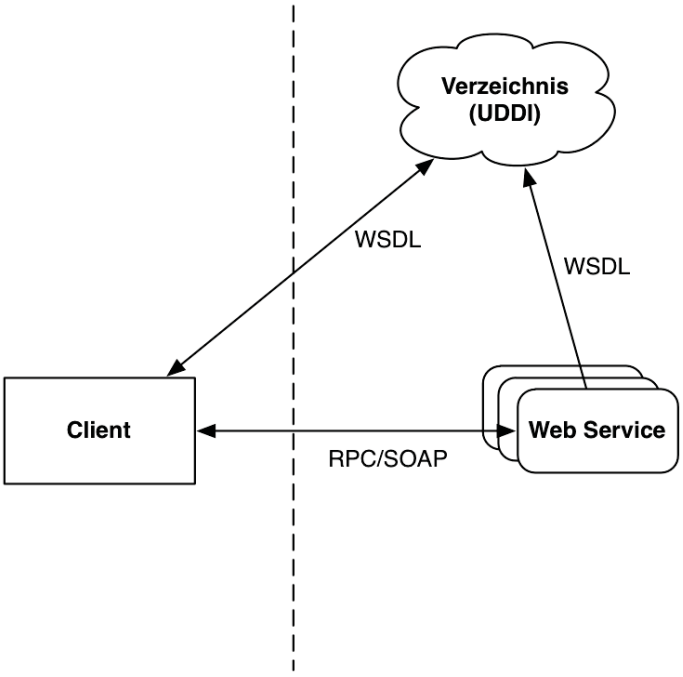

Abbildung 3.2.: Architektur einer Web Service Umgebung

sierte Verarbeitung erleichtert. Über entsprechende Spracherweiterungen, wie z.B. semantische Annotationen für WSDL (SAWSDL) [FL06], ist es darüber hinaus möglich, semantische Informationen über einen Dienst mit diesem zu verknüpfen. Die funktionale Implementierung eines Dienstes ist von der WSDL Beschreibung gänzlich entkoppelt und unterliegt prinzipiell keinerlei Einschränkungen hinsichtlich der verwendeten Programmiersprache oder Zielplattform. Im Wesentlichen können Web Services auf zwei grundlegend verschiedene Arten von den aufrufenden Clients adressiert werden: **RPC-basiert** (s.o.), wobei die Schnittstellenbeschreibung genaue Signaturen der vom Dienst bereitgestellten Funktionalitäten enthält, oder **Nachrichten-basiert**. Im letzteren Fall definiert die Schnittstelle eine Sammlung von Funktionen, welche auf unterschiedliche Eingabenachrichten reagieren können. Vom Client wird kein dedizierter Funktionsaufruf abgesetzt, sondern ein Nachrichtendokument via SOAP verschickt. Der zur Kommunikation verwendete zentrale Endpunkt eines Web Service ordnet dabei die Nachricht einer entsprechenden Aktion zu, welche serverseitig ausgeführt wird. Trotz aller Unterschiede lassen sich Web Services somit RPC konform einsetzen und daraus folgend nicht eindeutig von dieser Technologie abgrenzen.
Prinzipiell kann der funktionale Code für einen Web Service über ein angepasstes Framework aus der WSDL automatisiert erzeugt werden (contract-first). Ebenso ist es möglich, zunächst den Quellcode eines Dienstes zu erstellen und ausgewählte Funktionen generisch bereitzustellen, wobei die WSDL Beschreibung von

dem entsprechenden Framework erzeugt wird (code-first). Solcherlei Frameworks existieren in vielen verschiedenen Programmiersprachen und binden die abstrakte Definition eines Dienstes an die konkreten Gegebenheiten der verwendeten Zielsprache. Populäre Binding-Frameworks zur Verwendung innerhalb von Java sind beispielsweise JAX-WS[3], AXIS[4] oder AXIS2[5].
Zur Datenübertragung können Web Services de facto auf beliebigen Übertragungsprotokollen aufsetzen. I.d.R. kommt HTTP zum Einsatz, was eine hohe Zuverlässigkeit und Stabilität gerade in Bezug auf abgesicherte Netzwerke gewährleistet. Reine RPC Lösungen wie Java RMI oder CORBA stellen den Benutzer an dieser Stelle oftmals vor Probleme in Bezug auf Firewalls und gesperrte Nachrichten-Ports. Ebenso ist es möglich, die Benutzung spezieller Übertragungsprotokolle hinsichtlich des verfolgten Einsatzzweckes zu optimieren. Beispielsweise kann für die asynchrone Kommunikation SMTP [Pos82] oder FTP [PR85] zum Übertragen großer Nachrichten verwendet werden. Letzteres stellt jedoch generell ein Problem in Bezug auf die Effizienz und Performanz von Web Services dar, da alle Nachrichten und zu übertragende Binärdaten innerhalb eines XML Dokuments serialisiert werden müssen und im allgemeinen zu einem immensen Overhead führen. Ebenso ist der Sicherheitsaspekt in Bezug auf die Übertragung sensibler Daten in Form textueller XML Nachrichten erwähnenswert. Hier müssen ggf. entsprechende Mittel zur Verschlüsselung oder Signierung eingesetzt werden.
Trotz oder gerade wegen der hohen Flexibilität hinsichtlich der unterschiedlichen Kommunikationsaspekte und Freiheitsgrade beim Einsatz von Web Services, ist die Bereitstellung und Nutzung häufig sehr komplex und fehleranfällig. Insbesondere bedarf es erfahrene Nutzer, um in der Praxis immer wieder auftretende Inkompatibilitäten zwischen Diensten und korrespondierenden Client Aufrufen zu erkennen oder zu vermeiden. Das erforderliche technische Know-how liegt dabei deutlich über dem, welches z.B. beim Einsatz von reinen RPC Technologien benötigt wird.

3.3. Representational State Transfer

Oft als Unterkategorie von Web Services und Alternative zu RPC- und SOAP-basierter Kommunikation dargestellt, soll das Konzept des Representational State Transfer (REST) [Fie00] hier gesondert Beachtung finden. Zur besseren Abgrenzung von Standard Web Services, welche im vorherigen Abschnitt 3.2 vorgestellt wurden, werden Web-basierte Dienste, welche nach den Richtlinien von REST gestaltet sind im folgenden als REST Services bezeichnet.
Das Konzept des Representational State Transfer stellt in seinem Kern die Funk-

[3] `http://jax-ws.java.net/`
[4] `http://ws.apache.org/axis/`
[5] `http://ws.apache.org/axis2/`

tion des World Wide Web „wie es sein sollte“ dar und gilt heutzutage als Richtlinie für die flexible Nutzung von Web Standards. Oftmals werden auch einfache HTTP basierte Schnittstellen, welche ohne zusätzliche Transportschichten wie SOAP, etc. auskommen mit dem Begriff gekennzeichnet. Generell wird jede Funktionalität und jede Ressource eines Dienstes über einen eindeutigen Bezeichner, bzw. Unique Ressource Identifier (URI), adressiert und i.d.R. über HTTP zugänglich gemacht. Daraus folgt direkt, dass als Operatoren zum Aufruf einer Ressource u.a. die durch HTTP definierten POST, GET, PUT und DELETE als Kernfunktionalitäten zur Verfügung stehen. REST Services stellen somit eine Art Rückbesinnung auf einfache technologische Konzepte aus den Anfängen des World Wide Web dar, ohne sich in spezifischen technischen Besonderheiten und Zusätzen zu verlieren. Daraus ergibt sich in erster Instanz eine flexible und sehr einfach zugängliche Möglichkeit, verteilte Dienste plattformunabhängig zu adressieren. Die Zugriffe auf eine serverseitige Dienstressource durch den Client erfolgen dabei zustandslos, d.h. der Client besitzt lediglich das Wissen des Anwendungszustands aus dessen Kontext heraus der Aufruf initiiert wurde. Alle Informationen zum Zustand einer Ressource werden vom Server verdeckt und lediglich über die zugreifbare Repräsentation, üblicherweise im HTML oder XML Format, ausgedrückt. Im Gegensatz z.B. zu RPC-basierten Techniken hält ein REST konformer Service somit zu keinem Zeitpunkt Informationen über den Zustand der Client Anwendung, beispielsweise innerhalb einer HTTP Session, vor. Die triviale Art des strikten Zugriffs auf eine Dienstfunktionalität über Ihre URI und entsprechende Parameter führt wie oben beschrieben zu einer einfachen und flexiblen Nutzung der bereitgestellten Funktionalitäten. Da das Konzept allerdings keine standardisierte Schnittstellenbeschreibung der Dienste vorsieht, werden Informationen zur Nutzung von Ressourcen oftmals auf unterschiedliche Art und Weise propagiert. Dies kann in Form einer strukturierten Beschreibung auf Basis von z.B. XML erfolgen und wird beispielsweise bei der REST konformen Umsetzung von Standard Web Services realisiert. Im allgemeinen obliegt die Beschreibung der Schnittstelle allerdings einer natürlichsprachlichen Dokumentation der entsprechenden Dienste. Eine automatisierte Verarbeitung von REST Services ist daher nur bedingt möglich. Populäre Beispiele für REST konforme Web Dienste sind z.B. die Google Applikationen (Maps[6], Search[7], etc.) oder auch die LastFM API[8]. Zur Nutzung REST basierter Standard Web Services existieren zudem entsprechende Frameworks wie z.B. JAX-RS, bzw. dessen Referenzimplementierung *Jersey*[9] (vgl. Web Service Frameworks in Abschnitt 3.2)

[6] `http://maps.google.com`
[7] `http://www.google.com`
[8] `http://www.lastfm.de/api`
[9] `http://jersey.java.net/`

3.4. Proprietäre Remote Dienste

Trotz der Existenz einer Reihe gängiger und akzeptierter Standardverfahren und -technologien zur Bereitstellung und Nutzung verteilter Dienste werden zumeist von kommerziellen Software-/Serviceanbietern proprietäre Kommunikationsformate und Dienstimplementierungen eingesetzt und verbreitet. Dies ist zumeist der Unterstützung bestimmter spezialisierter Funktionalitäten oder der Kundenbindung an einen Hersteller oder ein Produkt geschuldet. Als populäres Beispiel ist hier beispielsweise die Integration von SAP ERP Funktionalitäten[10] über die sog. SAP Java Connector Middleware (SAP JCo) [SAP12b] zu nennen. Diese ermöglicht die Kommunikation zwischen SAP eigenen Advanced Business Application Programming (ABAP) [SAP12a] Implementierungen und Java Programmen. Die Nutzung solcher Dienste bringt normalerweise einen erheblichen Aufwand mit sich und ist durch reine Domänen- bzw. Anwendungsexperten kaum zu realisieren. Die automatische Integration auf Basis struktureller Schnittstelleninformationen ist zudem i.d.R., wenn überhaupt unterstützt, auf den Rahmen spezieller Ausführungsumgebungen beschränkt.

[10] `http://www.sap.com/erp`

4. Die jABC Modellierungsumgebung

Auf Basis der Evaluation verschiedener Modellierungsumgebungen in Abschnitt 2 fiel die Entscheidung der Basistechnologie für die zu erarbeitende Remote Service Umgebung auf das jABC, welches im folgenden grob erläutert wird.
Die jABC Entwicklungsumgebung wurde am Lehrstuhl für Programmiersysteme der Technischen Universität Dortmund mit Hinblick auf die grundlegenden Anforderungen einer intuitiv bedien- und leicht erlernbaren Plattform zur serviceorientierten Modellierung von Prozessen entworfen und implementiert und erfüllt dabei die Anforderung, diese Konzepte jedem Nutzer so weit wie möglich zugänglich zu machen. Die Entwicklung stand dabei ganz im Zeichen von „komplex für wenige - einfach für viele“ was in Wahrheit genau die Utopie beschreibt, die einer SOA genügen sollte. Jeder Nutzer mit einem Verständnis für die Programmierung von Computer Systemen ist generell in der Lage, durch die Anwendung verschiedener geeigneter Technologien eine Lösung zu einem angemessen spezifizierten Problem in einem akzeptablen Zeitrahmen zu implementieren und zu testen. Werden die Probleme aber komplexer, wie z.B. bei hoch skalierbaren Unternehmensprozessen und -applikationen, so werden Benutzer ohne klassisches IT Expertenwissen schnell an Ihre Grenzen stoßen. In einem solchen Fall ist es sicherlich möglich, dass entsprechende Domänenexperten, IT Berater und technisches Personal eng zusammenarbeiten; dies führt aber unweigerlich zu einer Situation in der Anwendungsspezifikationen in einer technische Sichtweise übersetzt werden müssen, was sich i.d.R. als zeitaufwändig und teuer herausstellt. Der strategische Fokus der jABC Modellierungsumgebung besteht darin, einfach zu benutzende, atomare Dienste für Domänenexperten, oder grob gesagt, Nutzer ohne tiefergehende IT Kenntnisse bereitzustellen und sie somit in eine Lage zu versetzen, in der es ihnen ermöglicht wird, konkrete Probleme Ihres Fachbereichs eigenständig und schneller zu lösen, indem eigene Ansätze direkt technisch umgesetzt werden können. Der Schwerpunkt liegt dabei auf der Abstraktion, der Verifikation und der Qualitätssicherung der zu verwendenden Dienste. Idealerweise sollten somit selbst komplexeste Unternehmenslösungen ähnlich einfach wie Bausätze aus LEGO Steinen oder anderen modularen Konstruktionssystemen zu handhaben sein. Mit einer entsprechend mächtigen Bibliothek an Service Komponenten (die LEGO Steine) ist der Modellierer in der Lage, selbst große Systeme aus entspre-

chend grundlegenden Einzelteilen sukzessive aufzubauen. Sollten derlei Komponenten nicht vorhanden sein (es fehlt ein LEGO Stein) kann statt einer Änderung und Überarbeitung der Anforderungen ein entsprechender Domänenexperte oder Dienstleister in Anspruch genommen werden, welcher die benötigte Funktonalität bereitzustellen im Stande ist (Einkaufen neuer Bausteine).
Die jABC Modellierungsumgebung stellt sich, betrachtet man die reine Kernfunktionalität, als grafisches Werkzeug zur Orchestrierung von lokal verfügbaren Diensten dar. Jegliche Funktionalität in Bezug auf die Verifikation, Ausführung, etc. wird darüber hinaus additiv modular durch ein Plugin Konzept realisiert, was den Vorteil einer flexiblen Konfigurier- und Erweiterbarkeit mit sich bringt.

4.1. Dienstrepräsentation

Innerhalb des jABC werden Dienste als sogenannte *Service Independent Building Blocks* (SIB) repräsentiert, welche über taxonomische Spezifikationen zu einer flexiblen Bibliothek von Diensten gruppiert werden können. Mit Hilfe der SIBs als atomare Komponenten werden komplexe Strukturen als *Service Logic Graph* (SLG) modelliert (vgl. Abschnitt 2.1), welche den Prozessfluss definieren. Jeder SIB hat dabei eine Anzahl fester oder frei konfigurierbarer Ein- und Ausgabeparameter sowie eine Menge von benannten und gerichteten Ausgangskanten welche sich in ihrem Verhalten nach den Ausgaben eines jeweiligen SIBs in Bezug auf die Ausführung der definierten Funktionalität des Dienstes richten. Somit kann ein SLG als eine Symbiose aus (gelabeltem) Transitionssystem (LTS) [BK08] und Kripke Struktur (KS) [Kri63, BK08] angesehen werden und bildet per Definition ein Kripke Transitionssystem (KTS) (vgl. Abschnitt 2.1). Das KTS, bzw. der SLG definiert den Kontrollfluss einer modellierten Applikation und kann seinerseits über ein Hierarchiekonzept wiederum als SIB zur Verfügung gestellt werden.
Typischerweise wird die Implementierung eines SIB in zwei Teile gesplittet, welche die konkreten Service Komponenten bilden: Ein SIB Container und ein SIB Adapter. Die komplette Dienstfunktionalität ist dabei im SIB Adapter enthalten, wohingegen der Container als reine Kommunikationsschnittstelle dient. Somit ist es u.a. möglich, korrelierende Dienstfunktionen in einem einzigen Adapter unterzubringen, welcher dann über verschiedene Container zugegriffen werden und somit unterschiedliche SIBs bedienen kann. Wie in Abb. 4.1 dargestellt, ruft der SIB, bzw. der Container lediglich die entsprechende Service Routine innerhalb des ihm zugeteilten Adapters auf. Atomare SIBs sind somit im klassischen Sinne eigenständig lauffähige Programme.
Dieses Konzept ist allerdings für den Endbenutzer komplett verdeckt, welcher sich einzig und allein auf die Benutzung verschiedener Bausteine stützen kann, die nach außen als einheitlich gekapselte und autonome Einheiten wirken. Lediglich der Anbieter der Dienste benötigt entsprechendes Know-how, um die jeweiligen

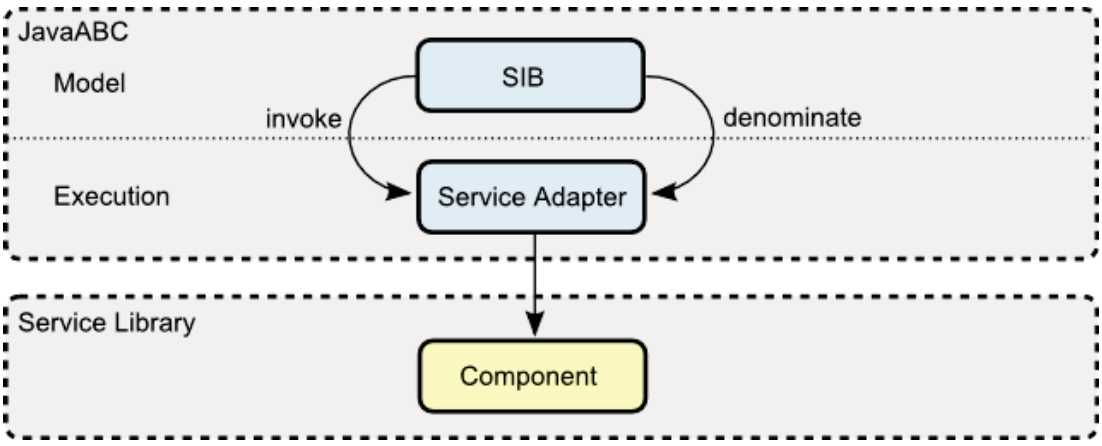

Abbildung 4.1.: Das Konzept der SIB Adapter.

SIBs auf Code Ebene zu implementieren, sollten sie nicht als komplexe Hierarchie aus anderen Bausteinen zusammengesetzt werden.
Wie oben beschrieben, ergibt sich der eigentliche Funktionsumfang des jABC über eine Anzahl von entsprechenden Plugin Komponenten. Technisch gesehen erfordert jede dieser Komponenten die Erfüllung bestimmter Anforderungen durch die SIBs, sollen diese in die Lage versetzt werden die entsprechenden Funktionalitäten nutzen zu können. Beispielsweise wird zur Laufzeitausführung einer Service Komponente eine entsprechende Routine vorausgesetzt, die von dem zuständigen Plugin abgearbeitet werden kann. Diese Spezifikationen werden durch Schnittstellenbeschreibungen innerhalb der verwendeten Programmiersprache Java definiert. Jeder SIB und jeder Adapter bezieht sich nun intern auf eine zugrundeliegende Java Klasse, welche die Implementierungen der geforderten Schnittstellen beinhalten muss, damit der SIB entsprechende Funktionsangebote der Plugins nutzen kann. Hierzu zählen insbesondere die Verifikation, die Ausführung oder auch die Code Generierung des finalen Modells.
Basierend auf dem Konzept der SIB Implementierungen ist neben den Plugins für die reinen Funktionalitäten der Modellierungsumgebung die Anzahl der bereitgestellten Service Komponenten entscheidend für die Flexibilität und Funktionsvielfalt der zu modellierenden Applikationen. Einer der Hauptpfeiler einer serviceorientierten Architektur ist dabei das Konzept der Wiederverwendbarkeit von atomaren oder komplexen Diensten. Durch den Aufbau und die Erweiterung der SIB Bibliothek lässt sich somit der Funktionsumfang und Einsatzbereich der Modellierungsumgebung ständig erweitern. Durch Einschränkungen der Bibliothek und der zur Verfügung stehenden Menge an Plugins kann die Software zudem domänenspezifisch einschränkt werden, was fachspezifischen Benutzergruppen einen deutlich fokussierteren Zugang ermöglicht.
Wie in Abschnitt 1 erwähnt, liegt die Gewichtung dieser Arbeit im folgenden auf

der flexiblen und umfangreichen Erweiterung und Anbindung der Service Bibliothek an diverse Arten von externen Diensten, bzw. der Bereitstellung externer Funktionalitäten als nutzbare Services. Hierbei wird verstärkt auch die Tatsache in den Fokus gerückt, dass bestehende Standards zur Bereitstellung und Nutzung externer Services zumeist Kenntnisse auf Code Ebene voraussetzen. Diese werden durch eine ganzheitliche Remote Service Umgebung weitestgehend entbehrlich gemacht und geben dem Nutzer neben der reinen Orchestrierung bereits existierender Bausteine auch die Möglichkeit, weiterführende Dienste einzubinden oder als Komponenten der Community zur Verfügung zu stellen.

4.2. Extreme Model Driven Development

Die Modellierung von Prozessen im jABC basiert im Wesentlichen auf dem Konzept des Extreme Model Driven Development (XMDD) [MS08, MS04], welches als ein serviceorientierter, modellbasierter Entwicklungsansatz über eine Reihe entscheidender Vorteile, gerade in Bezug auf die Integration von verteilten Service Komponenten verfügt:

- **Einfachheit**. Das jABC ist in erster Linie an Applikations-, bzw. Domänenexperten gerichtet, welche typischerweise keine oder nur wenige Programmierkenntnisse vorweisen können. Immer wieder konnte bei der Durchführung von diversen Projekten beobachtet werden, dass Benutzer die grundlegenden Ideen des Modellierungsprozesses in weniger als einer Stunde erfassen konnten [Nag09, doc08].
- **Agilität**. In Erwartung dauerhafter Modifikationen von Anforderungen und Modellen während der Entwicklungszeit von Prozessen unterstützt das jABC solcherlei Evolution, indem es dieses Vorgehen als Standard Herangehensweise ansieht und fördert.
- **Anpassbarkeit**. Die Bausteine, aus denen die Modelle bestehen, können völlig frei benannt und restrukturiert werden, um den Anforderungen der jeweiligen Domäne gerecht zu werden.
- **Konsistenz**. Der komplette Modellierungsprozess von der Erstellung erster Prototypen bis hin zur finalen Ausführung unterliegt ein und demselben Paradigma. Dies garantiert zu jeder Zeit eine Rückverfolgbarkeit sowie die semantische Konsistenz des aktuellen Modells.
- **Verifikation**. Mit Technologien wie Model Checking [CGP00] oder lokalen Prüfroutinen innerhalb der Service Komponenten unterstützt das jABC die konsistente Modifikation von Modellen. Die zugrundeliegende Idee ist dabei, dass lokale oder globale Eigenschaften formal spezifiziert werden, welche

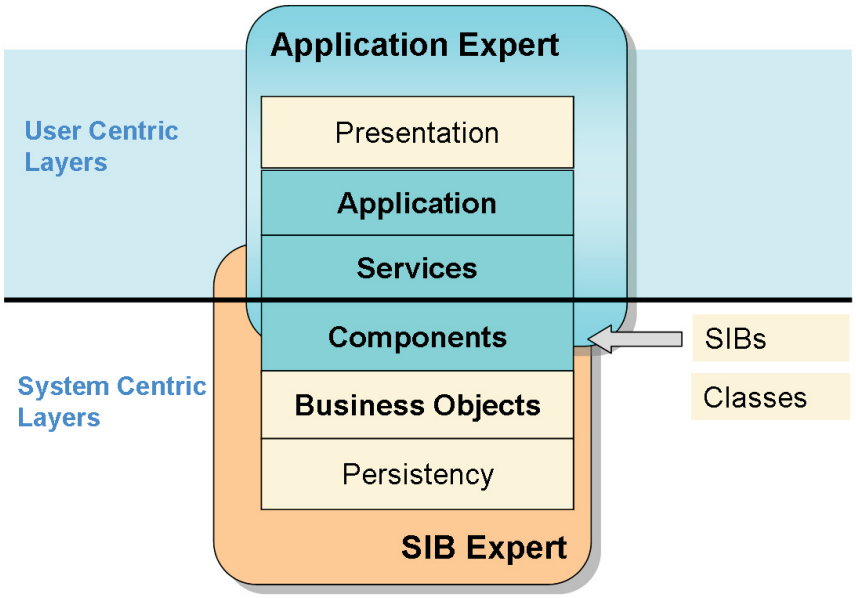

Abbildung 4.2.: Layered Architecture of jABC Applications

der Prozess zu jeder Zeit erfüllen muss. Die Spezifikationen werden dabei während der Modellierung zur Laufzeit der Umgebung ständig überprüft.

- **Serviceorientierung**. Bestehende Funktionalitäten können jederzeit auf einfache Art und Weise in das jABC integriert werden, indem sie innerhalb von SIBs gekapselt und somit als Bausteine verfügbar gemacht werden.
- **Ausführbarkeit**. Ein Modell kann verschiedenste Arten von ausführbarem Code enthalten. Dazu gehören abstrakte textuelle Beschreibungen (z.B. in der ersten Ausführungsphase eines Prototyps) oder aber konkrete Anweisungen innerhalb der finalen Implementierung.
- **Universalität**. Durch den Einsatz der Java Technologie als plattformunabhängige, objektorientierte Programmiersprache kann das jABC sehr einfach in verschiedensten technischen Kontexten oder Anwendungsdomänen eingesetzt werden.

Die grundlegende Idee hinter XMDD ist die Integration einer zusätzlichen Koordinationsschicht in die bekannte Drei-Schicht-Architektur klassischer, objektorientierter Programmierung. Diese Koordinationsschicht grenzt dabei die Applikations- und Serviceschicht deutlich voneinander ab, was in Abb. 4.2 hervorgehoben ist. Die Koordinationsschicht ist eine rein modellgetriebene Ebene, welche i.d.R. mit Hilfe eines grafischen Modellierungswerkzeuges erstellt und verwaltet wird.

Für gewöhnlich arbeiten zwei sich unterstützende Gruppen von Nutzern an einem XMDD Standardmodell:

- **SIB Experten**, bzw. Softwareentwickler, welche detailliertes Wissen über die Implementierung von SIBs und die Plugin Schnittstellen besitzen sowie
- **Applikationsexperten**, welche detailliertes Wissen über den zu modellierenden Prozess oder die Applikation besitzen, aber oftmals keinem technischen Hintergrund entstammen.

Wie in Abb. 4.2 dargestellt, modellieren Applikationsexperten die Geschäftslogik einer Anwendung mit Hilfe von zur Verfügung stehenden SIBs, welche Grundfunktionalitäten einer Domäne abbilden. Zudem stehen spezielle Komponenten als Platzhalter zur Verfügung, welche in frühen Phasen der Modellierung eingesetzt werden können, um noch nicht implementierte Funktionsteile zu simulieren. Sollten neue Bausteine innerhalb eines Prozesses benötigt werden, so kann der Applikationsexperte im ersten Schritt den Namen, zu verwendende Parameter und ausgehende Kanten innerhalb des jABC werkzeuggestützt spezifizieren. Die Implementierung und Ausarbeitung der eigentlichen Funktionalität wird in einem nächsten Schritt auf Basis dieser Spezifikationen in Zusammenarbeit mit dem SIB Experten realisiert.

Teil II.

Die jETI Remote Integrationsumgebung

5. Die jETI Remote Integrationsumgebung

Die in Abschnitt 4 vorgestellte jABC Modellierungsumgebung bietet dem Nutzer die Möglichkeit, lokal verfügbare Dienste aus einer Bibliothek auszuwählen und zu komplexen Prozessen und Applikationen zu orchestrieren. Neue Funktionen werden unter Berücksichtigung des SIB Konzepts in elementaren Bausteinen gekapselt und anschließend in die Sammlung der vorhandenen Komponenten integriert. Prinzipiell stellt die Nutzung von verteilten Diensten einen Spezialfall einer neuen Funktionalität dar. Global betrachtet liegt der zu benutzende Dienst auf Seiten des Anbieters (Server) dabei nicht im unmittelbaren Zugriffsbereich des Nutzers, was dazu führt, das eine lokale Schnittstelle zur Benutzung des Remote Services eingerichtet werden muss (Client). Diese kann anschließend innerhalb eines Service Independent Building Block gekapselt, mit zusätzlichen (semantischen) Informationen angereichert und dem Nutzer zur Verfügung gestellt werden. Bei der manuellen Integration von Remote Diensten im Zusammenspiel von Applikations- und SIB Experten sind somit folgende Arbeitsschritte durchzuführen:

1. Identifizierung des Dienstes, der verwendeten Remote Technologie und der Schnittstellenspezifikation.
2. Implementierung der lokalen Schnittstelle unter Verwendung der vom Dienst geforderten und optional angebotenen Parameter.
3. Kapselung der Schnittstelle innerhalb eines SIB und Bereitstellen der Komponente innerhalb der Service Bibliothek.
4. Einheitliche Annotation von Zusatzinformationen zur Dienstbeschreibung im Kontext des Semantic Web

Soll ein modellierter SLG oder auch ein anderweitig lokal vorhandener Dienst überdies als Remote Service zur Verfügung gestellt werden, sind zusätzlich folgende Punkte zu beachten

5. Identifizierung einer adäquaten Remote Technologie hinsichtlich der Effizienz der eingesetzten Übertragungsprotokolle für den dienstspezifischen Anwendungsfall und der Integrationsmöglichkeit in das jABC.

6. Möglichst strukturelle Beschreibung der Diensteigenschaften im Kontext der in Punkt 5 ausgewählten Technologie.

5.1. Eletronic Tool Integration

Mit der Eletronic Tool Integration Platform (ETI) [SMB97, BMW97, MBK97, Mar05, SMN05] existierte bereits im Jahre 1995 in der Vorgängerversion des jABC, dem Agent Building Center (ABC) [MS04], eine Möglichkeit zur einfachen Integration und Bereitstellung entfernt verfügbarer Dienste und Werkzeuge auf Basis eines proprietären Kommunikationsprotokolls. Ursprünglich als Plattform zur Evaluation und einfachen Verfügbarkeit von Softwarekomponenten ohne aufwändige Installation und Konfiguration durch den Benutzer konzeptioniert ging es dabei in erster Linie um die reine Funktionalität der Einbindung entfernter Dienste und weniger um die breite Unterstützung von Standardtechnologien, welche sich in ihrer heutigen Form in den Anfangszeiten des World Wide Web und der serviceorientierten Denkweise ohnehin erst entwickeln mussten. Historisch betrachtet liegt die Entwicklung von ETI somit vor der konzeptuellen und flächendeckenden Einführung von Web Services oder dem REST Konzept, wie es heute bekannt ist. Auch wenn sich die pure Funktionalität der Anbindung von verteilten Diensten an das jABC prinzipiell durch heutige Web Service Ansätze in Teilen überholt hat und ersetzen ließe, diente sie als Grundlage für die entwickelte Remote Service Plattform, welche analog zur Namensgebung der Modellierungsumgebung mit jETI (Java Electronic Tool Integration) betitelt wurde. Die Rechtfertigung für diese Entscheidung fußt dabei grundsätzlich auf den folgenden Beobachtungen und Eigenschaften:

1. Der Web Service Ansatz ist für viele Einsatzzwecke zu komplex und universell, was eine Benutzung der Technologie, gerade für Domänen- und Anwendungsexperten erschwert. Andere Technologien sind zumeist hinsichtlich der gebotenen Funktionalitäten und strukturellen Schnittstellenbeschreibungen, nicht mächtig genug (vgl. Abschnitt 3). Dies betrifft nicht allein die Integration von Diensten innerhalb der Modellierungsumgebung, sondern insbesondere auch die Bereitstellung von Softwarewerkzeugen aller Art als Remote Service.

2. Durch den engen Bezug zum jABC und die hauptsächlich darauf ausgerichtete Konzeption und Entwicklung der Remote Service Umgebung ist es möglich, einen sehr hohen Grad der Integration der Plattform als solche innerhalb der Modellierungsumgebung zu gewährleisten. Dies betrifft im speziellen das Importieren, Exportieren und Auffinden von Diensten direkt aus dem jABC heraus.

3. Generell soll die jETI Umgebung als offenes Grundkonzept für die im Rahmen dieser Arbeit bereits durchgeführten und zukünftigen Erweiterungen hinsichtlich der einzusetzenden Remote Technologien fungieren. Ausgehende von einer einfach zu benutzenden Kernfunktionalität sollen auch andere Technologien eingesetzt werden können, was die Benutzung von etablierten Verfahren und die Auswahl je nach Anwendungsfall ermöglicht. jETI ist somit nicht als Konkurrenz zu heutigen Remote Service Technologien zu sehen, sondern eher als Basiskonzept zur Integration multipler Ansätze, mit Ihren individuellen Vor- und Nachteilen (best-of-breed).

5.2. Architektur der (j)ETI Umgebung

In diesem Abschnitt wird die Neukonzeption und Verbesserung der bereits vorhandenen Remote Architektur ausgehend von den vorhandenen Vorarbeiten und hinsichtlich der in Abschnitt 3 evaluierten Technologien beschrieben und die grundsätzliche Funktionsweise erläutert. Der sukzessive Ausbau der Plattform, die Erweiterung durch zusätzliche Komponenten und Funktionalitäten sowie die wissenschaftlich konzeptuelle Relevanz wird darauf folgend in Abschnitt 6 anhand von praxisnahen Fallstudien und auf Basis der Veröffentlichungen [KMFS06, MKSN06, KJMS09, MKS08, KMSN08, MBK+08, BCV+08, KMS+08, KVW+08, KMK+08, MMK+09] detailliert abgehandelt. Insbesondere die Wege zur Entscheidungsfindung, die Einordnung der Arbeit innerhalb der Community und die „Lessons Learned“ werden dort diskutiert.

5.2.1. Vorarbeiten: Funktionsweise der ETI Plattform

Wie bereits erwähnt, existierte mit ETI bereits eine funktionsfähige Remote Umgebung für die Nutzung verteilter Dienste innerhalb des ABC. ETI selbst war zum damaligen Zeitpunkt nicht als offene Plattform für verschiedene Remote Technologien konzipiert, sondern basierte auf einem eigenen Konzept. Dabei lag der Fokus auf der für den Dienstanbieter möglichst einfachen serverseitigen Bereitstellung seiner Werkzeuge (unabhängig von der Modellierungsumgebung) und der für den Nutzer clientseitigen Ausführung von Softwarewerkzeugen im Kontext einer serviceorientierten Architektur, d.h. innerhalb des ABC.
Die grundlegende Architektur der ETI Umgebung wurde zunächst analog zur Neukonzeption des ABC in Java reimplementiert, um eine flexible Ausgangsbasis für die zu entwickelnde Remote Service Plattform und die technologischen Voraussetzungen für eine Integration des Systems in die Modellierungsumgebung zu schaffen. Abb. 5.1 zeigt den konzeptuellen und technologischen Aufbau der portierten ETI Umgebung, welche analog zur Namensgebung des jABC bereits in

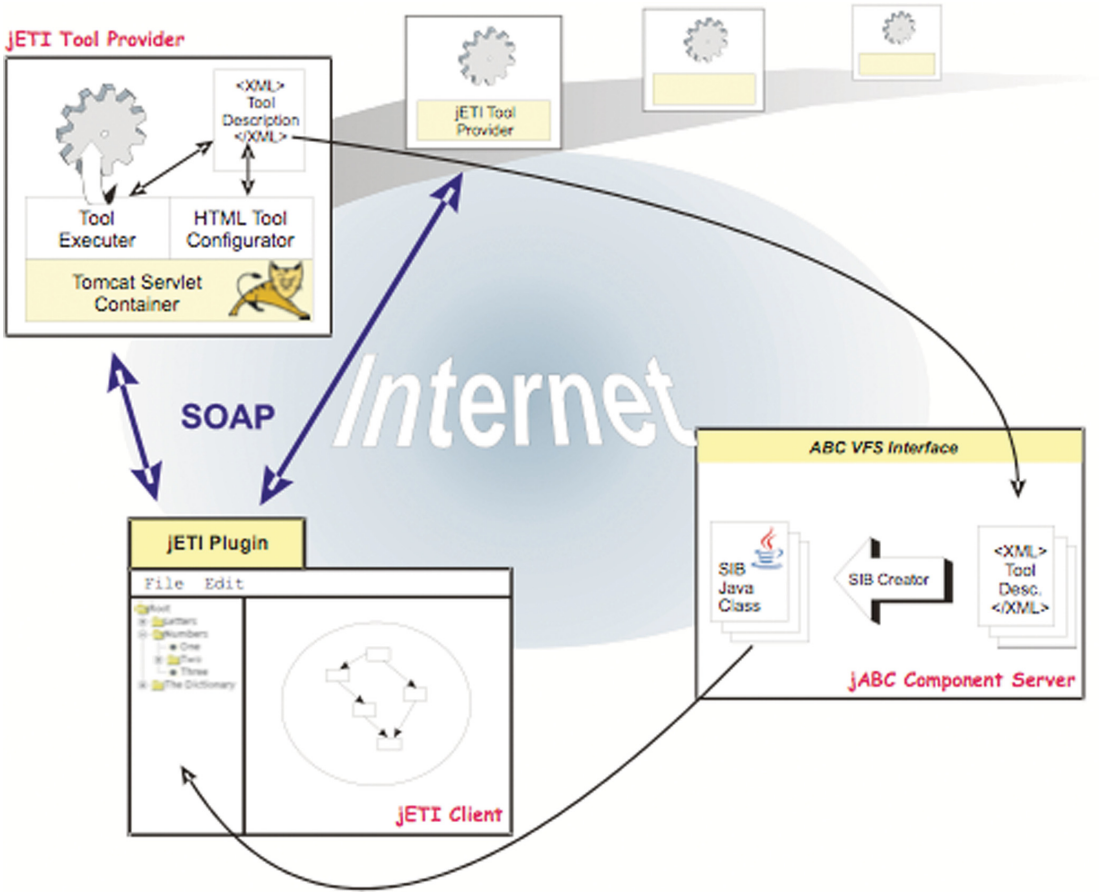

Abbildung 5.1.: (j)ETI Architektur und Funktionsweise

diesem Stadium in jETI [MNS05] umbenannt wurde. Wie im oberen, linken Teil der Abbildung zu erkennen, werden Softwarewerkzeuge, bzw. Dienstkomponenten angeboten, indem sie auf einem, bzw. mehreren unterschiedlichen Instanzen eines Toolservers bereitgestellt werden. Als Dienst kann dabei jedes ausführbare Programm auf Seiten des Servers dienen, welches sich parametrisiert oder skriptbasiert ansteuern lässt. Zudem kann technologisch bedingt auch direkt auf entsprechend freigegebene und auf dem Server vorhandene Java Bibliotheken zugegriffen werden. Ähnlich dem Konzept der Standard Web Services werden die zur Verfügung gestellten Dienste über ein strukturiertes XML Dokument beschrieben, welches generelle Informationen über den Service, als auch die technische Schnittstellenbeschreibung zum späteren Aufruf durch den Client enthält. Zur Erzeugung der Dienstbeschreibung existiert ein grafischer Konfigurator, der dem Anbieter als Hilfestellung zur Verfügung steht und ein intuitives Erstellen der XML Struktur ermöglicht. Auf Basis dieser strukturellen Beschreibung eines Service ist der sog. Komponentenserver (Abbildung unten rechts), welcher seiner-

seits Teil des Toolservers ist, in der Lage, komplette SIB Paletten zu generieren und über ein Web Interface für den Download bereitzustellen, was eine relativ einfache, wenn auch nicht komplett automatisierte Integration der Dienste in das jABC ermöglicht. Die SIBs enthalten dabei rudimentären Programmcode, welcher sich im Wesentlichen aus den der XML Beschreibung entnommenen Parametern und dem Identifikator des auszuführenden Dienstes zusammensetzt. Die zentrale Client Komponente ist über die Plugin-Schnittstelle des jABC in die Modellierungsumgebung integriert (Abbildung unten links).
Wie in Abschnitt 4 beschrieben setzt das Modularitäts- und Kompositionalitätsprinzip des jABC voraus, dass die SIBs zur Erfüllung von Funktionen im Kontext eines speziellen Plugins dessen vordefinierte Schnittstellen implementieren. So bedarf es beispielsweise zur lokalen Ausführung eines Service die entsprechenden Anweisungen an das *Tracer* Plugin [Doe06], die zentrale Ausführungskomponente des jABC. Die jETI Plugin Komponente nimmt hier eine Sonderstellung ein, indem sie zwar die entsprechenden Informationen zur Ausführung eines Dienstes vom SIB fordert (s.o.), aber nicht eigenständig, sondern während der Ausführung eines Modells durch den Tracer aufgerufen wird. Entsprechend identifizierte jETI SIBs werden von der Ausführungskomponente an den Client delegiert, welcher sodann für den konkreten Dienstaufruf verantwortlich ist.
Die Kommunikation zwischen Client und Server im Kontext der jETI Umgebung stützt sich dabei auf SOAP. Als XML basierte Kommunikationsschicht ist sie vor allem für den Austausch struktureller, textueller Daten geeignet. jETI selbst ist zum Austausch von Informationen zwischen der Modellierungsumgebung und dem Toolserver historisch bedingt jedoch auf ein dateibasiertes Verfahren angewiesen, da ETI ursprünglich entwickelt wurde, um Kommandozeilenwerkzeuge, welche im Wesentlichen auf Dateien arbeiten, auszuführen. Nachteilig ist an dieser Stelle also vor allem die Wahl der Kommunikationsschicht, welcher zum Transfer von Dateien eine textuelle Serialisierung aller Daten innerhalb der XML Struktur von SOAP notwendig macht. Performance- und Sicherheitseinbußen sind die unmittelbaren Folgen.
Weitere Diskrepanzen hinsichtlich einer konsistenten Bedienung dateiverarbeitender SIBs auf Seiten der Nutzer der Modellierungsumgebung bringt die interne Behandlung von Dateien bei der Ausführung von jETI Diensten mit sich. Hier setzt die Remote Umgebung auf ein virtuelles Dateisystem (VFS), so dass Daten erst mit einer Schreiboperation durch ein dediziertes SIB persistiert werden müssen. Analog dazu müssen persistente Daten zunächst eingelesen werden, bevor sie dem Toolserver zur Verfügung gestellt werden können. Neben der Tatsache, dass ein Nutzer der Remote Umgebung sich u.U. zunächst an das Konzept und die Funktionsweise des VFS einstellen muss, während lokale SIBs weiterhin direkt auf dem physischen Dateisystem des Hosts-Systems arbeiten, bringt das Arbeiten mit virtuellen Dateirepräsentationen in Sonderfällen weitere Nachteile mit sich. Wird der Client während einer Operation unterbrochen sind alle bis dato nicht persistierten Daten verloren. Insbesondere im Kontext von langlebigen Transak-

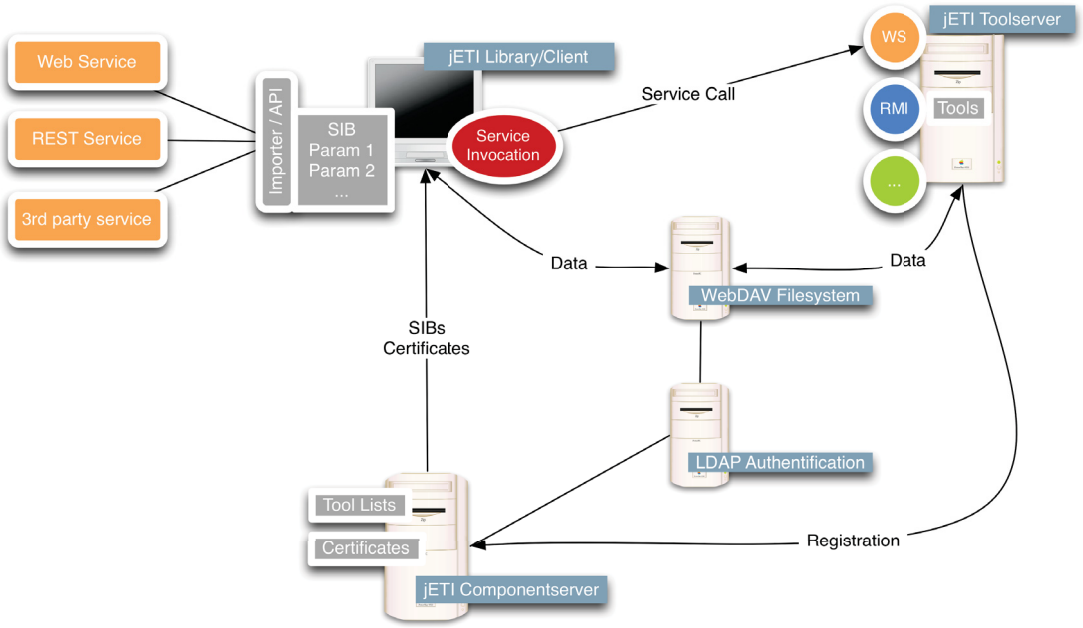

Abbildung 5.2.: Die neu konzeptionierte jETI Architektur

tionen (LRT) ist somit eine stabile und konsistente Ausführung eines Dienstes nicht immer gewährleistet.
Trotz der Einschränkungen und konzeptuellen Stolpersteine stellt die existierende (j)ETI Umgebung jedoch eine gute Ausgangsbasis für die letztendlich zu entwickelnde Remote Service Umgebung dar, welche im folgenden erläutert wird.

5.2.2. Restrukturierung und Neukonzeption der Architektur

Ausgehend von der durch jETI bereitgestellten Basistechnologie und -konzeption bestanden die ersten Schritte der Restrukturierung in der Öffnung und Flexibilisierung der Architektur. Des Weiteren sollte das komplette Framework deutlich modularer gestaltet, die Integration in die Modellierungsumgebung verbessert und eine konsistente Methode zur Annotation semantischer Informationen geschaffen werden. Resultierend aus diesen Überlegungen zeigt Abb. 5.2 die neu erarbeitete jETI Architektur. Der jETI Toolserver, bzw. nunmehr und im weiteren Service Provider Site (SPS) genannt, bildet nach wie vor die Basiskomponente, wenn es um die Bereitstellung und Integration von auf dem Server lauffähigen Applikationen ohne grafische Oberfläche geht. Die Kernfunktionalitäten sind im Wesentlichen aus dem Vorgängersystem übernommen worden. Um die Anpassung an unterschiedliche Dienstanforderungen und die Modularität hinsichtlich der Schnittstellen zu verbessern (siehe Anforderungen A1 und A2) wurde der SPS eine

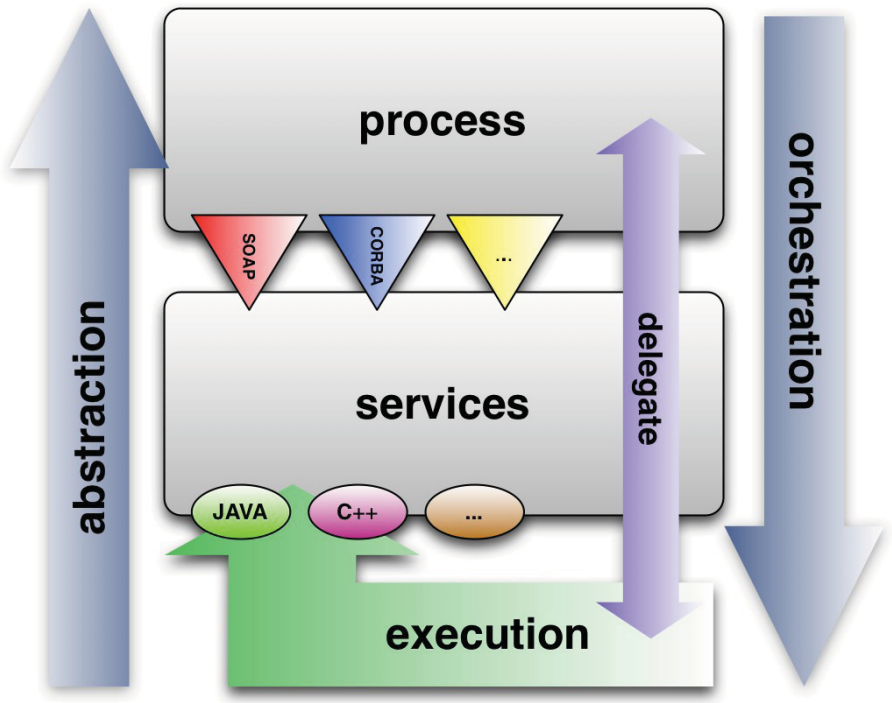

Abbildung 5.3.: Das Virtualisierungskonzept der Dienstausführung innerhalb der jETI Remote Service Umgebung

abstrakte Kommunikationsschicht hinzugefügt. Auszuführende Dienste kommunizieren nun nicht mehr ausschließlich über eine interne SOAP Verbindung, sondern werden nach außen als standardkonforme Service Komponenten propagiert. Eine offene API vereinfacht dabei die Integration neuer Protokolle (siehe Anforderung A6) von denen zunächst der vorherige Ansatz der SOAP Kommunikation als vollwertiger Web Service sowie eine RMI Variante über HTTP/Socket umgesetzt wurde. Abb. 5.3 verdeutlicht den Ansatz der Virtualisierung der Ausführungs- und Kommunikationsschicht innerhalb der Remote Service Umgebung. Dienste werden innerhalb von Arbeitsabläufen mit Hilfe der Modellierungsumgebung orchestriert und je nach Anwendungsfall und Service-Technologie über entsprechende Remote-Protokolle aufgerufen. Die Dienste selbst sind dabei gekapselt und von der funktionalen Implementierung und Ausführungsschicht entkoppelt. Der zur Dienstausführung notwendige Datenaustausch zwischen Client und Server über die Übertragung von Steuerungsparametern hinaus erfolgt im Gegensatz zur vorherigen Implementierung nicht mehr ausschließlich als serieller Strom von XML Daten, sondern bedient sich optional einer zusätzlichen Dateisystemkomponente auf Basis des Web-based Distributed Authoring and Versioning (WebDAV) Protokolls [web07]. Die Daten werden vom Server auf einem dedizierten physischen Dateisystem abgelegt und liegen über WebDAV ebenfalls im Zugriffsbereich des

Clients, der gerade bei großen Datenmengen auch zeitverzögerte und asynchrone Schreib/Lese Operationen durchführen kann. Dies unterstützt insbesondere die Ausführung von langlebigen Transaktionen und erhöht zudem die Sicherheit gegenüber Datenverlust während einer Dienstausführung (siehe Anforderung T2).
Eine weitere gravierende Neuerung bezieht sich auf die komplette Extraktion des Komponentenservers (CS) und die damit einhergehende Abgrenzung gegenüber der SPS. Der Komponentenserver übernimmt in dem neuen Szenario die Hauptfunktionalität der Kommunikationsschnittstelle zwischen der Modellierungsumgebung (Client) und den unterschiedlichen SPS, welche sich zuvor anmelden und dem CS bekannt gemacht werden müssen. Über verwaltete Listen sind dem CS sodann alle Dienste eines Universums der SPS bekannt, so dass er in der Lage ist, Anfragen von Clients hinsichtlich der Suche nach einem Service oder der Verfügbarkeit von Ausführungsressourcen effizient zu beantworten (vgl. Anforderung S4). Eine Bündelung oder das Clustering redundant verfügbarer Dienste ist ebenso inbegriffen. Eine Zertifikatsverwaltung regelt zudem die kontrollierten und restriktiven Aufrufe eines Dienstes durch den Client. Als universelles Authentifizierungsmodul dient in der jETI Umgebung ein Lightweight Directory Access Protocol (LDAP) [WHK97] Server, welcher zudem auch die Rollen und Zugriffsrechte des WebDAV Dateisystems übernimmt. Eine Benutzerverwaltung kann somit zentralisiert über ein Single Sign On (SSO) Pattern erfolgen.
Clientseitig ist es gelungen, die Integration in die Modellierungsumgebung weiter zu erhöhen und somit die User Experience zu verbessern. Die ursprüngliche Ausführungskomponente auf Basis eines jABC Plugins wurde an die neuen Gegebenheiten auf Seiten der Server-Landschaft angepasst, und insbesondere durch einen neu konzeptionierten Service Browser zur Ansicht der von einem Komponentenserver verwalteten Dienste ergänzt. Somit ist es Benutzern des Systems möglich ohne den Umweg eines dedizierten Downloads über das Web Interface der SPS, bzw. des Toolservers einen Dienst (automatisiert) zu integrieren, was das gefühlt lokale Arbeiten mit Paletten von Remote Services deutlich verstärkt (vgl. Anforderung S2).
Die Integration von standardisierten Remote Technologien außerhalb des Einflusses der SPS wurde ebenfalls über entsprechende Client-Erweiterungen realisiert. So existieren verschiedene Integrationsmodule zur (teil)automatisierten Anbindung von Web Services, REST konformen Diensten und proprietären Schnittstellen von Drittanbietern (z.B. R-Project, SAP; siehe Abschnitt 6.4), welche direkt aus dem jABC heraus über das Client Plugin integriert werden können.
Die Erfüllung der Anforderungen T1 und T3 bzgl. der Anreicherung der integrierten Dienste mit semantischen Informationen und der damit u.a. ermöglichten automatischen Erzeugung von Prozessmodellen, wurde durch eine Schnittstelle zur Anbindung externen Synthesekomponenten realisiert. Somit war es möglich, beispielhaft verschiedene Ansätze zur Prozesssynthese auf Basis der im Umfeld meiner Arbeitsgruppe entwickelten Werkzeuge *PROPHETS* [Nau09] und *jMosel* [TWMS06], sowie mit *GEM* [KL07, Kai07] und einem auf der Programmierspra-

che *GOLOG* basierten Ansatz zwei externe Methoden zu integrieren. Die Wissensrepräsentation erfolgt dabei in Form von Datenflussannotationen (*use*, *gen*, *kill*) über die Ein- und Ausgabeinformationen eines integrierten Dienstes (Vorbedingung/Effekt). Durch im jABC als Service Logic Graph spezifizierte Typ- und Aktionstaxonomien ergibt sich anschließend in Kombination mit in Semantic Linear Time Logic (SLTL) [SMB97] spezifizierten temporallogischen Formeln das Domänenmodell als Grundlage für die automatische Generierung von Prozessen. Insbesondere die Probleme bei der Integration von SOAP basierten Web Services mit individuell typisierten Datenstrukturen konnte davon profitieren, da hier die Automatisierung weiter voran getrieben wurde. Die unterschiedlichen Methoden bilden dabei divergente Synthesekonzepte ab, welche die Flexibilität des Ansatzes unterstreichen sollen:

PROPHETS: Das Verfahren der *Process Realization and Optimization Platform using a Human-readable Expression of Temporal-logic Synthesis* (PROPHETS) erstellt auf Basis der Typ- und Aktionstaxonomien zunächst ein sog. Konfigurationsuniversum aller Möglichkeiten auf dessen Basis mittels Forward Proof Construction (Forward-Chaining) ein Tableau der Lösungswege erstellt wird. Durch die Evaluation der temporallogischen Spezifikationen werden dabei konkrete Lösungen auf dem Domänenmodell ermittelt. Prinzipiell führt der zu Grunde liegende Algorithmus eine Suche aus, welche durch die parallele Auswertung des Konfigurationsuniversums und der SLTL Formeln realisiert wird.

jMosel: Das jABC Plugin *jMosel* zur Anwendung von *Monadic Second-Order Logic on Strings* (M2L(Str) [Chu63]) berechnet mit Hilfe kompositioneller Automatenkonstruktion und auf Basis des Domänenmodells sowie der temporallogischen Spezifikation ein als deterministischer, endlicher Automat abgebildetes Produkt aller möglichen Lösungswege. Dabei ist besonders der kompositionelle Aspekt hervorzuheben, da *jMosel* zunächst Teilautomaten aus den atomaren Bestandteilen der formalen Spezifikation generiert und diese anschließend zusammenfügt.

GOLOG: Ausgehend von der Spezifikation der Typ- und Aktionstaxonomien wird zunächst ein, in der auf der *ALGOL*-Sprachfamilie [WH66] und dem Situationskalkül von McCarthy und Reiter [MH69, Rei91] basierenden Programmiersprache *GOLOG* verfasstes, Domänenmodell erstellt. Ein entsprechender Interpreter führt im Anschluss die Planung/Synthese der Lösungswege mit Hilfe von Backward-Chaining Mechanismen durch.

GEM: Wie auch bei der Integration des *GOLOG* Ansatzes wird bei Verwendung des agentenbasierten *Goal-oriented Enterprise Management* (GEM) zunächst ein Domänenmodell, in diesem Fall mit Hilfe der Programmiersprache PROLOG und dem Event-Kalkül von Kowalski und Sergot [KS86], erstellt. Auf diesem können im Anschluss einzelne "Agenten„ genannte Komponenten Teillösungen suchen, die letzten Endes via Backward-Chaining zu einem gewünschten Lösungsprozess führen.

Im Zuge der Integration unterschiedlicher Synthesekonzepte eröffnete sich zudem die Möglichkeit mit Hilfe der semantischen Beschreibungen das Auffinden von Diensten auf Basis Ihrer Merkmale und angebotenen Funktionalitäten zu flexibilisieren (vgl. Anforderung S4). Zu diesem Zweck wurde in Zusammenarbeit mit dem Lehrstuhl für Software Engineering der Universität Potsdam die in der Diplomarbeit [Kub05] beschriebene *miAamics* Personalisierungsumgebung angepasst und in Form eines Plugins in das Modellierungs-Framework integriert. Ursprünglich entwickelt, um komplexe Datenmengen von Angeboten auf eine große Anzahl von Benutzern abzubilden und dabei Echtzeitanforderungen zu genügen, nutzt miAamics priorisierte Regeln in Form boole'scher Formeln und algebraische Entscheidungsdiagramme (ADD) zur Ergebnisfindung auf einer durch Taxonomien abstrahierten Lösungsmenge. Die Integration und Anwendung der einzelnen Technologien sowie eine technisch detaillierte und ausgearbeitete Beschreibung des sukzessive erarbeiteten und hier vorgestellten Gesamtsystems innerhalb verschiedener Domänen und Kontexte ist dem folgenden Kapitel 6, bzw. den Veröffentlichungen [KMFS06, MKSN06, KJMS09, MKS08, KMSN08, MBK+08, BCV+08, KMS+08, KVW+08, KMK+08, MMK+09] zu entnehmen. Geprägt durch die unterschiedlichen Anforderungen während des Mitwirkens in verschiedenen Communities, Arbeits- und Forschungsgruppen konnte die Remote Service Umgebung zudem kontinuierlich angepasst, technologische Sackgassen frühzeitig erkannt und das Gesamtsystem somit letztendlich Schritt für Schritt verbessert werden. Insbesondere wird innerhalb der Veröffentlichungen im Detail auf die in der Praxis aufgetretenen Schwierigkeiten bzgl. der automatisierten und nicht-technischen Integration und Anbindung von Remote Diensten eingegangen und das Vorgehen anhand von Fallstudien diskutiert.

6. Fallstudien

In diesem Kapitel werden verschiedene, im Rahmen dieser Arbeit durchgeführte, Fallstudien beschrieben und in den Gesamtkontext eingeordnet. Die Entwicklung der Remote Service Umgebung und die zu Grunde liegenden Konzepte und Anforderungen sind dabei maßgeblich von den Erfahrungen und Ergebnissen der einzelnen Fallstudien und der aktiven praktischen Mitarbeit in den entsprechenden Communities beeinflusst. Der wissenschaftliche Aspekt dieser Arbeit, insbesondere der Entscheidungsprozess der letztendlich in der vorliegenden Remote Service Umgebung mündete, wird dabei anhand der Einzelszenarien und der verschiedenen Ansätze herausgestellt.
Die Fallstudien werden im folgenden beschrieben, jeweils grob zusammengefasst und individuell abschließend anhand einiger Zeilen zu den „Lessons learned“ in den Gesamtkontext des Systems eingeordnet. Anforderungen und Ziele werden dabei für jeden Anwendungsfall konkretisiert. Die detaillierten Ausarbeitungen der Fallstudien finden sich in den dieser Arbeit zugehörigen Veröffentlichungen [KMFS06, MKSN06, KJMS09, MKS08, KMSN08, MBK+08, BCV+08, KMS+08, KVW+08, KMK+08, MMK+09].

6.1. Statistische Auswertung in der Bioinformatik

Im Rahmen eines EU geförderten Projektes am Zentrum für Proteomik im BioMedizinZentrum Dortmund[1][2] wurde jETI bereits in einem frühen Stadium eingesetzt um die Interprozesskommunikation zwischen den einzelnen Arbeitsgruppen zu optimieren. Dies umfasste im Wesentlichen die Zusammenarbeit der Statistiker, welche verschiedene Analyseverfahren in Form von R-Skripten (siehe Abschnitt 6.4) bereitstellten, und Biologen, welche im Anschluss die entstandenen Werkzeuge für Ihre Auswertungen einsetzen konnten. Hier stellten sich insbesondere die Vorteile von jETI im Kontext eines nicht technischen Benutzerkreises heraus: Die Einzelkomponenten der statistischen Auswertungsverfahren wurden von den Statistikern als SIBs innerhalb von jETI/jABC bereitgestellt und konnten anschließend von den Biologen völlig frei und intuitiv eingesetzt sowie konfiguriert und orchestriert werden. Rücksprachen mit dem Fachbereich und individuelle

[1] `http://www.bmz-do.de`
[2] `http://www.zap-do.de`

Anforderungsdefinitionen der Nutzer bzgl. spezialisierter Prozessketten konnten so im Laufe des Projekts minimiert werden. Der Fokus für die Arbeit lag dabei zunächst auf der Optimierung der Benutzeroberfläche auf Seiten der Service Anbieter und User, welche durch Ihre Arbeit mit der Remote Umgebung wichtige Erkenntnisse in Bezug zur Nutzung durch Domänenexperten lieferten. Im Verlauf des Projekts kristallisierten sich jedoch zunehmend die noch etwas umständliche manuelle Kapselung der R-Skripte (vgl. Abschnitt 6.4 innerhalb von SIB Komponenten sowie die zu diesem Zeitpunkt synchrone Prozesskommunikation im Kontext von langlebigen Transaktionen als Probleme heraus. Hier konnten somit erste empirische Daten erhoben werden, welche sich im folgenden erheblich auf die Wizard gesteuerte, semi-automatische Integration von proprietären Diensten ohne standardkonforme Schnittstellen sowie die spätere Neukonzeption der Systemarchitektur mit Hinblick auf LRT auswirkten.
Die Arbeit im Rahmen des ZAP wird anhand einer beispielhaft durchgeführten Flüssigchromatographie mit Massenspektrometrie-Kopplung (LC/MS) in [KMFS06, MKS08] detailliert beschrieben. Darüberhinaus wurde durch die interdisziplinäre Zusammenarbeit die Entwicklung eines eigenständigen Bio-jETI[3] Zweigs motiviert, der insbesondere die Integration der EBI SOAPLab Web Services[4] aus dem EMBOSS Projekt[5] [RLB00, LNMS09, MKS08] zur Folge hatte und sich konzeptuell ausschließlich auf die Domäne der Bioinformatik konzentriert. Hier konnte in enger Zusammenarbeit mit den Anwendern praxisnahe Erfahrungen gesammelt werden, welche sich in der Umsetzung der automatisierten Integration von Web Services (siehe Abschnitt 5.2.2) widerspiegeln und diese stark motiviert haben.

6.2. Formale Methoden für kritische Systeme in der Industrie

Im Rahmen des europäischen Forschungskonsortiums für Informatik und Mathematik (ERCIM)[6] wurde mit Hilfe der jETI Remote Integrationsumgebung innerhalb der Arbeitsgruppe Formal Methods for Industrial Critical Systems (FMICS) ein europaweites Software Repository mit Partnern aus bisher sieben Ländern aufgebaut[7]. Der Schwerpunkt lag dabei auf der Integration von formalen Methoden zur Software Verifikation und deren Transfer und Ausbau mit dem Ziel einer industriellen Nutzung. Wie schon im Kontext der Fallstudie 6.1 lag auch hier der Fokus auf der manuellen Kapselung komplexer Dienste aus unterschiedlichen

[3] http://biojeti.cs.tu-dortmund.de
[4] http://www.ebi.ac.uk/soaplab
[5] http://emboss.sourceforge.net
[6] http://www.ercim.eu/
[7] http://www.inrialpes.fr/vasy/fmics

und hochspezialisierten Teilbereichen der einzelnen Standorte. Von Interesse im Sinne einer empirischen Untersuchung des Nutzungsverhaltens der Remote Service Umgebung war im Gegensatz zu 6.1 vor allem der technische Hintergrund der Teilnehmer und die Anzahl der verteilten Dienstanbieter. Im Rahmen von Workshops wurden einzelne Teilnehmer im Umgang mit dem System geschult und konnten im Anschluss ihre Software Werkzeuge eigenständig integrieren und der Community zu Nutze kommen lassen. Neben der weiteren Verbesserung der grafisch gestützten Integration von Diensten auf Seiten der Service Provider Site (SPS) wurden auch hier vor allem Schwächen des Prototypen hinsichtliche der Ausführung von LRT und dem redundanten Hosting von Werkzeugen auf unterschiedlichen SPS mit jeweils eigenen Komponentenservern deutlich, was deren spätere konzeptionelle Trennung beförderte (siehe Abb. 5.2). Auch wurden Grenzen der Remote Service Umgebung aufgezeigt, falls spezielle Tools sich nicht ohne weiteres über skriptbasierte Schnittstellen kapseln ließen. Dies zog Forschungen und Machbarkeitsstudien hinsichtlich der automatischen Fernsteuerung und Ausführung von rein GUI basierten Werkzeugen nach sich, die bis heute andauern und im Rahmen des Fazits und des Ausblicks in Abschnitt 8 gesondert betrachtet werden.
Die Arbeit im Rahmen der FMICS Community wird in [MKSN06] detailliert vorgestellt.

6.3. Semantic Web Services Challenge

Bereits in einem frühen prototypischen Entwicklungsstadium der Remote Service Umgebung wurde jETI im Rahmen der Semantic Web Services Challenge (SWSC)[8] eingesetzt. Ziel und Vision der bis heute andauernden und im Rahmen von regelmässigen Workshops stattfindenden Veranstaltung war und ist es, neue Methoden zur semantischen Annotation, Evaluation und Ausführung von (Web) Service Komponenten im Rahmen komplexer, serviceorientierter und businessrelevanter Problemstellungen zu bewerten und im Kontext der SWSC Community mit Hinblick auf das Semantic Web oder Web 3.0 zu vergleichen. Die Schwerpunkte lagen auf zwei Teilbereichen:

1. Die (automatisierte) Daten- und Prozesskommunikation zwischen autonomen Systemen (Mediation) sowie
2. die dynamische, automatische und flexible Auffindbarkeit von Diensten (Discovery),

welche durch entsprechend komplexe Szenarien evaluiert wurden. Die zu lösenden Problemstellungen entwickelten sich dabei über die teilnehmenden Arbeitsgrup-

[8] `http://www.sws-challenge.org`

pen kontinuierlich weiter und verlangten so insbesondere nach einer Evolution, bzw. Agilität der jeweiligen Lösungsansätze und Systeme. Eine detaillierte Beschreibung der Szenarien findet sich auf der Website der SWSC(siehe Fußnote). Im Gegensatz zu den Fallstudien aus 6.1 und 6.2 bedeutete dies weniger die empirische Untersuchung des Nutzerverhaltens und die daraus resultierende Weiterentwicklung der Remote Service Umgebung hinsichtlich Usability und User Experience als vielmehr die Erweiterung und Anpassung von jETI in Bezug zu neuen technologischen Anforderungen und Möglichkeiten, insbesondere der automatisierten Prozesssynthese. Des Weiteren konnte die Arbeit im Vergleich mit diversen Forschungsgruppen aus der Domäne der Semantic Web Services eingeordnet und mit verschiedenen Ansätzen verglichen werden (siehe [BCV+08, KVW+08] und Abschnitt 7).

6.3.1. Mediation

Im Rahmen der Problemstellung der SWSC Mediation galt es für die teilnehmenden Arbeitsgruppen eine Kommunikationsschicht zwischen zwei autonomen Systemen zu entwickeln, deren Verhaltensbeschreibung als Flussgraph, bzw. durch die RosettaNet Spezifikation[9] gegeben war. Komponenten der jeweiligen Systeme wurden als Web Services von den Organisatoren der Challenge zur Verfügung gestellt. Die Evaluation der Lösungen erfolgte auf Basis des positiven Datenaustausches zwischen den Systemen gemäß der Spezifikation.
Zum Zeitpunkt des Einstiegs mit jETI und Beginn der SWSC im Jahre 2006 befand sich die Remote Service Umgebung in ihrer damals ersten, reimplementierten Version, welche sich im Wesentlichen auf die Urfunktionalitäten von ETI beschränkte und somit noch keinerlei technisch ausgearbeitete Anbindung von Web Services ermöglichte. Über eine erste manuelle Kapselung von WS Aufrufen innerhalb von SIBs wurde im Laufe der Challenge ein Verfahren entwickelt, um standardisierte, d.h. über ein Web Service Description Language (WSDL) Dokument beschriebene Web Services möglichst automatisiert und ohne technisches Vorwissen über jETI in die Modellierungsumgebung zu integrieren [KMSN08, KJMS09]. Hier stellte sich vor allem die innerhalb der Community vielfach hervorgehobene Universalität und die daraus resultierende Komplexität von Web Services als Problem heraus und verdeutlichte auf drastische Art und Weise die Existenzberechtigung auch restriktiver Remote Technologien, wie sie jETI von Grund auf zu Leisten vermag (vgl. Abschnitt 5). Je nach Anwendungskontext ist der hohe Freiheitsgrad von Web Services demnach nicht notwendig und verkompliziert i.d.R. die Arbeit mit verteilten Service Komponenten auf Seiten der Dienstanbieter und -nutzer. Die technische Hürde wird dabei extrem inkrementiert, was den bis heute stark technisch geprägten Einsatz der Technologie erklärt und unterstreicht.

[9] `http://www.rosettanet.org`

Dennoch ist es gelungen mit Hilfe von Einschränkungen auf Seiten der Modellierungsumgebung eine vollautomatische Integration von universellen Web Services über jETI zu realisieren. Insbesondere die Definition komplexer Datentypen und Strukturen, welche von den Dienstanbietern innerhalb der Web Service Beschreibungssprache WSDL vorgenommen werden können, stellten dabei ein Problem dar, welches auf Basis von Restriktionen auf der Ebene der Modellierung gelöst werden konnte ohne den Funktionsumfang von Web Services einzuschränken [KJMS09].
Nach wie vor besteht die Lösungs-Utopie der SWSC in einer vollständig deskriptiven Spezifikation eines Systems, welches sich auf dieser Basis selbstständig an Veränderungen von außen anzupassen vermag. Hier ist es den einzelnen Teilnehmern erst durch mehr oder weniger starke Restriktionen hinsichtlich der semantischen Spezifikation von Web Services gelungen, das Mediationsproblem der SWSC zu bearbeiten und letztendlich automatisiert zu lösen. jETI bedient sich dabei einer einheitlichen Schnittstellenspezifikation in Kombination konzeptuell verschiedener Ansätze zur Prozesssynthese (vgl. Abschnitt 5.2.2). Die Integrative und interdisziplinäre Entwicklung, bzw. Erweiterung von einigen der integrierten Synthese Komponenten (*PROPHETS*, *jMosel*) durch kooperierende Arbeitsgruppen an der TU Dortmund erlaubte es, die für jETI und die SWSC benötigten Anforderungen im Detail umzusetzen [MBK+08].Da eines der erklärten Ziele der SWSC der Vergleich und die Kollaboration der einzelnen Teilnehmer untereinander beinhaltete, wurde über die fachbereichseigenen Methoden hinaus in enger Zusammenarbeit mit der Forschungsgruppe um M. Kaiser und J. Lemcke (SAP Research) ein weiteres Verfahren zur automatischen Lösung des Mediationsproblems experimentell in die Remote Service Umgebung und das Modellierungs-Framework integriert (*GEM*) und überdies an einer zusätzlichen, *GOLOG* basierten Integration des Lehrstuhls für Software Engineering der Universität Potsdam mitgearbeitet.
Die komplette Arbeit an der Lösung des Mediationsproblems im Rahmen der Semantic Web Services Challenge sowie die erarbeiteten Erfolge und aufgetretenen Probleme auf dem Wege dorthin, insbesondere bei der Umsetzung der Syntheseansätze im Kontext ihres praktischen Einsatzes mit jETI, sind im Detail in den Veröffentlichungen [KMSN08, MBK+08, KMK+08, MMK+09] beschrieben.

6.3.2. Discovery

Im Teilbereich der Auffindung von Services lieferte die SWSC ein komplexes Problemszenario in dessen Kontext ein passender Dienst über eine Vielzahl unterschiedlicher, inkonsistenter und nicht formaler Parameter effizient gefunden werden sollte. Ein adäquates Auswertungsverfahren über eine triviale Servicesuche auf Basis von Verzeichnisdiensten hinaus war an dieser Stelle eine Grundvoraussetzung zur Erarbeitung eines Lösungsansatzes. Da im Rahmen der Diplomarbeit zur miAamics Personalisierungsumgebung [Kub05] mit einem auf algebraischen

Entscheidungsdiagrammen aufbauenden Verfahren zur Personalisierung erarbeitet wurde, wurde im Kontext der SWSC der Versuch unternommen, die Technologie entsprechend angepasst zur Lösung des Discovery Problems einzusetzen. In Teilen gelang es dabei mit Hilfe von priorisierten Regeln einzelne Dienste entsprechend den Spezifikationen zu identifizieren, jedoch zeigten sich mit steigender Komplexitität der Problembeschreibung die Grenzen des Systems. Dennoch konnte hier der Grundstein zur Erweiterung des einfachen Mechanismus zur Auffindung von Diensten im Rahmen der Remote Service Umgebung gelegt werden. Die Arbeit am Discovery Problem der SWSC ist in [KMS+08] konkret beschrieben.

6.4. Anbindung verteilter Dienste mit proprietären Schnittstellen

Neben der Unterstützung von standardisierten Remote-Technologien bestand eine weitere Anforderung an jETI darin, die möglichst einfache Integration von Diensten mit proprietären Schnittstellen von Drittanbietern zu ermöglichen (vgl. Anf. 1.1 und 1.1.1). Wie auch schon in den Fallstudien 6.3 oder 6.1 wurden erste Anwendungen in der Praxis über die manuelle Kapselung von Services innerhalb von SIBs durchgeführt. Dies führte zwar zu einem kurzfristigen Aufbau einer geeigneten Service Bibliothek, setzte aber fundierte Kenntnisse der jeweiligen Dienstschnittstelle und ein Verständnis für die programmiertechnische Umsetzung der Service-Anbindung voraus.
In diversen kleineren Projekten, wie z.B. dem Einbinden von SAP/R3 Diensten über den SAP Java Connector (SAP JCo) wurde die Integration solcher Dienste untersucht und die Möglichkeiten einer Wizard-gestützten Methodik abgewogen. Da i.d.R. die herstellereigenen Schnittstellen keinerlei Standardkonformität aufweisen, sind auch Hilfskomponenten zur teilautomatisierten Anbindung der Dienste durch den Benutzer auf ein eng gefasstes Universum von Services beschränkt und an dieser Stelle hochspezialisiert. Dennoch bieten sie dem Nutzer die Möglichkeit und Freiheit aus einer eingegrenzten Domäne beliebige Dienste komfortabel und ohne technisches Know-how innerhalb des Modellierungs-Framework zu verwenden. Auf diesen Erfahrungen aufbauend wurde in der Diplomarbeit [Mus09] die Integration von Diensten des R-Projekts (vgl. Fallstudie 6.1) automatisiert und in der Praxis angewendet.
Unter dem Label *Bio-jETI* wurde darüberhinaus eine neue Community geschaffen, welche die Services verschiedener Dienstanbieter aus dem Bereich der Bioinformatik vereint und dabei Gebrauch sowohl von jETIs Möglichkeit der automatisierten Anbindung von standardisierten Web Services als auch der

Wizard-gestützten Integration von Diensten diverser Anbieter (u.a. BioMoby[10], R-Project[11]) macht.
Details zum praktisch erprobten Einsatz von jETI im biotechnologischen Umfeld finden sich neben den Angaben auf der in den Fußnoten angegebenen Website des Bio-jETI Projekts insbesondere in [MKS08].

6.5. Heterogene Service Mashups

Im Zuge des Social Web, bzw. Web 2.0, ist im Laufe der vergangenen Jahre eine Vielzahl von allgemeinen und spezialisierten sozialen Netzwerken entstanden. Diese bieten ihren Nutzern aktiv die Möglichkeit, semantisch rudimentär annotierte Inhalte bereitzustellen, Kontakte zu knüpfen und somit komplexe Informationsstrukturen aufzubauen. Um die technische Integration, Akzeptanz und Verbreitung der jeweiligen Produkte, bzw. Dienste dabei zu steigern sind viele Anbieter solcher Netzwerke und Consumer Services dazu übergegangen, mehr oder weniger komplexe und strukturierte Schnittstellen zur Verfügung zu stellen. Mit deren Hilfe sind die Benutzer in der Lage, völlig neue Anwendungen auf Basis der einzelnen Dienste zu kreieren und der Community wieder zur Verfügung zu stellen. Das Prinzip dieser sog. Mashups ist dabei die „Serviceorientierung für die breite Masse“. Neben einfachen Modellierungs-Frameworks, wie z.B. Yahoo Pipes[12], die sich in erster Linie an Laien und Hobby User wenden, können jedoch mit komplexeren Modellierungswerkzeugen und der zunehmenden Bedeutung der Anbieter, Menge an verfügbaren Daten und Vielfalt der angebotenen Services im World Wide Web auch ernstzunehmende Anwendungen umgesetzt werden.
Aufbauend auf den Erfahrungen aus den vorherigen Fallstudien wurde in einem ersten Proof of Concept (POC) zunächst versucht, populäre Dienste wie Google Maps[13], Amazon[14] oder Wikipedia[15] zu einem neuen, komplexeren Service zu verschmelzen. Technisch gesehen zeigte sich dabei das ganze heterogene Spektrum von aktuellen Remote Technologien, wie proprietäre und standardisierte RPC Aufrufe, Standard Web Services oder Dienste, die auf den Prinzipien des Representational State Transfer (REST) basieren [Fie00]. Insbesondere letztgenannte Technologie erfreut sich bei den Anbietern höchster Beliebtheit, da es mit relativ geringem Aufwand bzgl. Implementierung und Infrastruktur auf Seiten der Provider und der User ermöglicht wird, auch komplexe Funktionalitäten bereitzustellen, bzw. anzusteuern. Bei der Integration in die Remote Service Umge-

[10] `http://biomoby.org`
[11] `http://r-project.org`
[12] `http://pipes.yahoo.com`
[13] `http://maps.google.com`
[14] `http://www.amazon.com`
[15] `http://www.wikipedia.org`

bung wirkte sich dabei jedoch nachteilig aus, dass REST Services i.d.R. keinerlei strukturierten oder formelle Beschreibung der Schnittstellen liefern und lediglich auf natürlichsprachlich dokumentieren APIs aufbauen. Lediglich der technische Aufruf über einen *Uniform Resource Identifier* (URI), bzw. HTTP bilden einen gemeinsamen Nenner aller Dienste. Um eine möglichst automatisierte Integration von REST Services zu gewährleisten wurde jETI, analog zu der in Fallstudie 6.4 beschriebenen Integration von proprietären Diensten, um eine Wizard-basierte Lösung erweitert, die es dem Nutzer ermöglicht, über eine minimale Anzahl von Interaktionen mit dem System einen Dienst als SIB zu kapseln und anschließend innerhalb des Modellierungs-Framework einzusetzen. Dabei werden Informationen, wie Parameter, Service Endpunkte und das Ein-, bzw. Ausgabeformat eines Dienstes abgefragt, welche zu einem vollständigen Client für den jeweiligen Service kombiniert werden.
Ein erster POC beinhaltete die Kombination einer Geo Lokalisierung mit Hilfe der Dienste von InstaMapper[16] und Google Maps. Zu den jeweiligen Standorten wurden anschließend Points of Interest (POI) in der näheren Umgebung über Wikipedia ausgelesen, entsprechende Fotos zu den Standorten über Flickr[17] angezeigt und mit Produkten aus dem Bestand von Amazon kombiniert. Anschließend wurde über einen Telekommunikationsservice des Berliner Fraunhofer Instituts für offene Kommunikationssysteme (FOKUS)[18] eine SMS mit den entsprechenden Informationen an eine beliebige Nummer versendet.
Die Service Integration sowie die Durchführung des POC ist im Detail in [KJMS09] beschrieben. Darüberhinaus ist in einem weiterführenden Projekt mit dem Fraunhofer Institut (s.o.) der POC zu einer vollständigen Applikation zur Notfallalarmierung im Kontext des Assistant Living ausgebaut und im Rahmen des internationalen IMS Workshops 2008 präsentiert worden [Kub08].

[16] `http://www.instamapper.com`
[17] `http://www.flickr.com`
[18] `http://www.fokus.fraunhofer.de`

7. Verwandte Arbeiten

Um die verschiedenen Aspekte der Arbeit in den wissenschaftlichen Gesamtkontext einordnen zu können und kritisch zu hinterfragen, werden im folgenden diverse Forschungsprojekte und technologisch ausgereifte (Standard)-Konzepte diskutiert. Um dabei eine bessere Vergleichsmöglichkeit zu schaffen, werden einzelne Teilaspekte und Anforderungen der Arbeit verschiedenen, logisch getrennten Ansätzen gegenübergestellt, da eine ganzheitliche Betrachtungsweise aufgrund der verwendeten Methodik und des damit verbundenen Konzeptes von jETI zu einer Verzerrung der Ergebnisse führen würde. Generell betrachtet und vom konzeptuellen Ansatz ausgehend adressiert die konzipierte Remote Service Umgebung Applikations- und Domänenexperten ohne Programmierkenntnisse oder all zu stark ausgeprägtes technisches Vorwissen. Unter diesem Aspekt sollen die im folgenden betrachteten Arbeiten in Bezug zu jETI gesetzt werden.
Im Kontext serviceorientierter Architekturen spaltet sich die Integration von Remote Services konzeptuell zunächst grob in zwei Teilbereiche:

1. Die durch Generierungswerkzeuge gestützte Integration der Komponenten und anschließende manuelle (grafische) Orchestrierung auf syntaktischer Ebene, und

2. die automatische Aggregation relevanter Dienste zu komplexen Prozessmodellen auf Basis deskriptiver Informationen auf semantischer Ebene.

Darauf Bezug nehmend werden in Abschnitt 7.1 zunächst verwandte Arbeiten aus dem erstgenannten Teilbereich mit jETI verglichen, wohingegen Abschnitt 7.2 die automatische Generierung von Prozessen auf Basis semantischer Informationen beleuchtet (vgl. auch Abschnitt 6.3). Weitere, in Bezug auf die o.a. Trennung untergeordnete und somit teilbereichsübergreifende Technologien, werden in den Abschnitten 7.3 - 7.5 beschrieben. So sind weiterführende Ansätze zur effektiven und teils automatisierten Suche von Diensten (Discovery) in Abschnitt 7.3 zu finden. Abschnitt 7.4 konzentriert sich auf Web-basierte Technologien die insbesondere im Kontext des konzeptionierten jETI-Testbed (vgl. Abschnitt 9) verglichen werden. Die eher einfach gehaltene Ausprägung von heterogener Serviceorchestrierung im Sinne sog. Mashups wird in Abschnitt 7.5 abschließend verglichen.

7.1. Modellgetriebene Entwicklung und Dienstintegration

In diesem Abschnitt werden verschiedene Modellierungsumgebungen und technologische Konzepte hinsichtlich Ihrer Fähigkeiten zum Umgang mit Remote Services betrachtet. Insbesondere werden solche Frameworks betrachtet, die eine möglichst automatisierte und intuitive Integration domänenübergreifender Dienste bereitstellen. Prinzipiell ist es mit jedweder Art von Modellierungsumgebung, welche beliebige Service-Implementierungen zulässt, möglich, Remote Services manuell zu integrieren und anschließend zu orchestrieren. Da der Schwerpunkt hier jedoch auf der möglichst intuitiven, automatisierten und programmierungsfreien Integration von Remote Diensten (vornehmlich Web Services) liegt, werden ausschließlich Frameworks und Konzepte zu einem Vergleich herangezogen, welche einen nicht trivialen und den Benutzer unterstützenden Ansatz verfolgen. Eine vollständige Deckungsgleichheit mit den evaluierten Frameworks aus Abschnitt 2 ist daher weder gewünscht noch gegeben.

7.1.1. Seaside

Das Seaside Projekt[1] [BDR08] zur Komposition und Modellierung des Kontrollflusses von dynamischen Web-Applikationen unterscheidet sich zunächst grundlegend durch seinen strikt Code-basierten Ansatz von der Arbeit im Kontext der grafischen Modellierung mit Hilfe des jABC. Zustandssensitive Komponenten werden als zusammenhängender, einfacher Source Code zu komplexen Web-Applikationen aggregiert, wobei jeder Baustein einen minimalen Teil einer Website kapselt. Unterstützend bietet das Seaside Framework eine API zur Anbindung gängiger Web 2.0 Technologien wie asynchronem Javascript und XML (AJAX) [Gar05] oder Comet [Gra10, CM08]. Die Modellierung von nicht Web basierten Applikationen spielt an dieser Stelle nur eine untergeordnete bis gar keine Rolle. Um die anvisierte Zielgruppe von Seaside, die Web Entwickler, bei der Implementierung von Applikationen zu unterstützen existieren eine Reihe von Hilfsmitteln, wie z.B. eine Debugging Komponente, vergleichbar mit dem Tracer Plugin [Doe06] des jABC. Ebenfalls existiert die Möglichkeit, dass User Interface einer Anwendung als Baumstruktur individueller Komponenten abzubilden und durch Konzepte wie Zustandsabbildungen und Speicherung der Kontrollflussausführungen spezielle Anforderungen an Web-Applikationen abzubilden. So ist es bspw. möglich, zu jeder Zeit zu einem bereits durchlaufenen Zustand zurückzukehren und den „Zurück“-Button des Web Browsers zu verwenden ohne den Kontrollfluss der Applikation zu durchbrechen (*snapshot*). Ebenfalls können Applikationsaus-

[1] `http://www.seaside.st`

führungen jederzeit pausiert und später wieder aufgenommen werden (*continuation*), womit sich asynchrones Verhalten effizient modellieren lässt. Durch die strikte Fokussierung auf Web-Applikationen und deren spezieller Anforderungen weißt Seaside an dieser Stelle mitunter deutliche Vorteile gegenüber der Orchestrierung mit Hilfe des jABC auf, welches sich bspw. im Rahmen des Designs von UI Komponenten als eher ungeeignet herausgestellt hat. Die Kontrollflussmodellierung innerhalb von Seaside wird in Smalltalk[2] abgebildet, die etablierte Anbieter entsprechender Entwicklungsumgebungen, wie. z.B. Cincom[3] zur Integration des Frameworks bewogen hat. Die finale Applikation wird als XHTML Generat erzeugt. Mit Hilfe des universellen Code Generator Plugin [Joe11] des jABC können im Vergleich dazu beliebige Ausgabeformate aus der modellierten Applikation generiert werden.

7.1.2. Taverna

Die Taverna Modellierungsumgebung, welche bereits in Abschnitt 2.3 vorgestellt wurde, setzt zur Integration von Remote Diensten ebenso wie jABC/jETI auf ein modulares Plugin Konzept. Es existieren eine Reihe von mehr oder weniger generischen Integrationserweiterungen, die es ermöglichen, vorgegebene domänenspezifische Web Service Bibliotheken oder beliebige Remote Dienste innerhalb der modellierten Prozesse zu verwenden. Erstgenanntes wird bspw. über das BioCatalogue Plugin[4] realisiert, welches den Zugriff auf ein Verzeichnis von Life Science Web Services ermöglicht. Eine Integration neuer Dienste ist nur dann möglich, wenn diese innerhalb des Repositories registriert und angeboten werden. Werden semantisch annotierte Web Services verwendet, kann das SADI Plugin[5] eingesetzt werden, welches über Standardtechnologien die Kommunikation mit Semantic Web Services ermöglicht. Einen generischeren Ansatz verfolgt die REST API von Taverna [tav12], welche es erlaubt, beliebige Dienste über GET, POST, PUT und DELETE Anfragen anzusprechen. Vorlagenbasierte Konfigurationen erlauben die Kommunikation mit mit beliebigen Ein- und Ausgaben.
Sollen mit Taverna modellierte Prozesse als Web Service bereitgestellt werden, so kann das OPAL Plugin[6] verwendet werden. Mit dessen Hilfe kann eine Applikation gekapselt und als SOAP-Style Web Service nach außen propagiert werden.
Alles in allem ist der von Taverna verfolgte Ansatz dem von jETI recht ähnlich, wobei letzteres den Anspruch einer ganzheitlichen Integration von Remote Diensten definiert. Die Taverna Plugins sind an dieser Stelle deutlich atomarer und lagern die unterschiedlichen Technologieansätze in separate Plugins aus. Eine

[2] `http://www.smalltalk.org`
[3] `http://www.cincomsmalltalk.com/main/products/visualworks/`
[4] `http://www.biocatalogue.org/`
[5] `http://sadiframework.org`
[6] `http://www.nbcr.net/software/opal/`

generische Anbindung beliebiger SOAP-Style Web Services, wie jETI sie bietet, existiert momentan nicht, was aber durch die eher domänenspezifische Akzeptanz von Taverna innerhalb der Bioinformatik und den daher beschränkten Raum der verfügbaren Anwendungen zu erklären ist.

7.1.3. Konzeptuelle Arbeiten

Die hier vorgestellten Arbeiten beschreiben keine vollständigen Modellierungsumgebungen und können daher nur zu einem konzeptuellen Vergleich mit jABC/jETI herangezogen werden. Auf dem Gebiet der Modellierung von Remote Diensten, bzw. Web Services zählen dabei insbesondere die Veröffentlichungen von Achilleos, Kapitsaki und Papadopoulos [AKP11] sowie Xiao und Zheng.
[XZ07]
Achilleos führt in erster Linie eine Methode zur Verwendung des Eclipse Modeling Framework (EMF)[7] ein, welche auf PML Modellen basiert (Presentation Modeling Language) [Ach10]. In Kombination mit einem WSDL Modell zur Beschreibung der Web Services und entsprechenden Tools zur automatischen Generierung von Code Fragmenten aus diesen XML Spezifikationen entwickelte er mit seiner Gruppe entsprechende Code Generatoren, die für unterschiedliche Zielplattformen lauffähige Applikationen mit grafischer Benutzeroberfläche erzeugen können. Die Modellierung kann in grafischer Form mit entsprechenden Tools des Eclipse Framework erfolgen, jedoch sind manuelle Schritte auf technischer Ebene erforderlich, um die jeweiligen Web Services zu integrieren. Hierzu zählt insbesondere die Generierung von clientseitigen Code-Fragmenten zur Kommunikation mit den Diensten. Im Gegenzug fußt der Ansatz auf etablierten und anerkannten Technologien im Umfeld der Softwareentwicklung (Eclipse IDE, EMF, PML) und hat daher gerade im fundierten technischen Umfeld der SOA Integration hohe Relevanz.
Xiao und Zheng hingegen nutzen Flussdiagramme zur Erzeugung von Service Modellen mit abstrakten Dienstpräsentationen als Petri Netz [Pet62] mit Hilfe des Web Service Kalküls. Die Petri Netze werden in einem weiteren Transformationsschritt über einen Discovery Mechanismus in Modelle mit konkreten Web Services überführt (grounding). Daraus resultierend wird ein BPEL Prozess abgeleitet, welcher in einer entsprechenden Ausführungsumgebung, wie z.B. Microsoft BizTalk[8] ausgeführt und evaluiert werden kann. Hier liegt der praktische Nutzen eher in der Verwendung abstrakter Business Modelle, welche eher lose spezifiziert sein können. Konkretisiert wird der Prozess erst bei der Überführung in die entsprechenden Petri Netze mit Hilfe des Kalküls. jABC/jETI setzen hier von vornherein auf ein semantisch abgeschlossenes Prozessmodell, welches direkt aus-

[7] `http://www.eclipse.org/modeling/emf/`
[8] `http://www.microsoft.com/germany/biztalk/default.mspx`

führbar ist und konkrete Dienste beinhaltet, die automatisiert integriert werden können. Die abstrakte Repräsentation von Remote Services mit einem automatisierten Grounding-Mechanismus über entsprechende Discovery Routinen existiert zum Teil über die auch schon in der Fallstudie der Semantic Web Services Challenge (vgl. Abschnitt 6.3) angewandten Methoden zur Prozesssynthese.

7.2. Semantic Web Services

In diesem Abschnitt wird die hauptsächlich durch die Teilnahme an der Semantic Web Services Challenge (vgl. Abschnitt 6.3) motivierte und in diesem Kontext explizit erforderliche semantische Annotation von Diensten sowie die daraus abzuleitende automatisierte Integration und Orchestrierung von Remote Service Komponenten betrachtet. Dem Best-of-Breed Ansatz folgend (vgl. Abschnitt 1) wird hier diesbzgl. insbesondere die Kombination von jABC/jETI und der momentan umgesetzten Integration unterschiedlicher Synthese-Frameworks herangezogen (vgl. Abschnitt 5.2.2).

7.2.1. ServiceComposer

Der 2008 von W. Binder et al vorgestellte *ServiceComposer* [BCFJ08] dient der automatisierten Komposition von Web Services als BPEL4WS Modell [bpe02] in Bezug zu benutzerspezifischen, funktionalen Anforderungsdefinitionen. Die verwendeten Dienste werden dabei über ein Verzeichnis von Service-Beschreibungen in einer dedizierten Query-Sprache angefragt und semantisch durch einfache Methodensignaturen oder komplexe Wissensrepräsentationen in Form von OWL-S [M$^+$04] oder WSMO (vgl. Abschnitt 7.2.2) Ontologien beschrieben. Basierend auf den entsprechenden Annotationen nutzt der ServiceComposer AI-Planning Ansätze und Optimierungsalgorithmen zur bestmöglichen Bestimmung einer gewünschten Dienstfunktionalität und somit zum Auffinden geeigneter Service Komponenten. Da neben den rein funktional spezifizierten Anforderungen komplexe Nebenbedingungen auftreten können (hochspezialisierte Dienstaspekte werden nicht erfüllt, dedizierter Ausschluss von Komponenten aus Kosten- oder Verlässlichkeitsgründen, etc.) arbeitet der ServiceComposer nicht vollständig automatisiert und bedient sich während des Anfrageprozesses der Interaktion mit dem Benutzer. Durch die Berücksichtigung von partiellen Treffern bei der Dienstsuche, erhöht sich jedoch die Flexibilität in Bezug auf die zur Verfügung stehenden Services und die Wahrscheinlichkeit eines validen, letzten Endes dann allerdings weniger genauen, Ergebnisses erheblich [BCFJ08].
Im direkten Vergleich bietet jETI in Kombination mit der *PROPHETS*, *jMosel* und *GOLOG* basierten Prozesssynthese sowie dem agentengesteuerten Pla-

nen mit Hilfe von *GEM* momentan vier konzeptuell verschiedene Ansätze (vgl. Abschnitt 5.2.2). Service Beschreibungen werden in Form von Datenflussinformationen annotiert. Die Zuordnung erfolgt auf Basis proprietärer, jedoch im Kontext des jABC konsistenter Taxonomierepräsentationen als Service Logic Graph. Der eigentliche Syntheseprozess folgt (bislang) den Mechanismen des Forward- und Backward-Chaining, als auch automatentheoretischen Ansätzen. Der zur Zeit durch den Einsatz unterschiedlicher Synthesetechnologien bereits erreichte hohe Grad an Flexibilität hinsichtlich der Nutzung kann überdies durch die modulare Anbindung weiterer Methoden zur automatisierten Generierung von Prozessmodellen noch gesteigert werden.

7.2.2. Web Service Modeling Framework

Das von Fensel und Bussler am STI International (vormals Digital Enterprise Research Institute (DERI))[9] als ganzheitliches Konzept entwickelte *Web Service Modeling Framework* (WSMF) [FB02] hat den Anspruch an eine vollständige Umgebung zur Beschreibung verschiedener Aspekte bezogen auf (Semantic) Web Services. Basierend auf Arbeiten von Leymann und seinem Vorschlag der *Web Service Flow Language* (WSFL) Beschreibungssprache [Ley01], bzw. Ankolenkar und dem semantischen Konzept zur Beschreibung von Diensten (*DAML-S* [A$^+$01]), identifizieren Fensel und Bussler zwei sich ergänzende Kernkonzepte, die als zentraler Dreh- und Angelpunkt des WSMF gelten und gleichzeitig die fundamentalen Anforderungen an eine Umgebung definieren:

1. Das starke **Entkoppeln** aller beteiligten Komponenten und
2. einen komplexen **Mediator-Dienst**, der es jeder Dienstrepräsentation erlaubt mit anderen Komponenten zu kommunizieren und dabei hoch skalierbar zu sein.

Die vier durch das WSMF propagierten Hauptelemente, um diese Anforderungen zu erfüllen sind dabei:

1. **Ontologien**, welche die formale Semantik für alle Komponenten bereitstellen und damit den Sprachraum, bzw. das Datenmodell über alle verwendeten Elemente aufspannen
2. **Web Services**, um die funktionalen und verhaltensorientierten Aspekte semantisch beschrieben in Form nicht-funktionaler Eigenschaften, Fähigkeiten und Schnittstellen eines Dienstes abzubilden
3. **Goals**, um die Nutzerperspektive und die Anforderungen an einen zu erzeugenden Prozess, bzw. einen aufzurufenden Dienst hinsichtlich seiner Funk-

[9] `http://www.sti2.org`

tionalität und Schnittstellen zu definieren und

4. **Mediatoren**, welche die Interoperabilität zwischen den verwendeten Elementen (Ontologie zu Ontologie, Web Service zu Goal, Goal zu Goal und Web Service zu Web Service) sicherstellen.

Basierend auf der konzeptuellen Grundlage des WSMF ist mit der *Web Service Modeling Ontology* (WSMO) [FLP+06, RKL+05] ein Modell erarbeitet worden, welches weiterhin mit dem *Web Service Execution Environment* (WSMX) [HCM+05, MMCZ06, VMK+07] in einer konkreten Referenzimplementierung resultiert.
Ausgehend von den Ideen und Methoden des WSMF, bzw. WSMO soll hier ein Vergleich im Kontext semantischer Annotationen von Web Services und der daraus resultierenden (teil)automatisierten, modellgetriebenen Softwareentwicklung bis hin zur Prozesssynthese angestrebt werden. Der praktische Ansatz bezieht sich dabei auf einen technologischen Mix konkreter Methoden und Werkzeuge, welche insbesondere im Rahmen der Semantic Web Services Challenge angewendet wurden und eine Möglichkeit der Umsetzung des durch WSMF definierten Konzeptes bilden.
Die identifizierten Hauptelemente werden im folgenden durch konkrete Technologien beschrieben und bilden in Ihrer Gesamtheit die Implementierung einer ganzheitlichen Semantic Web Service Umgebung nach WSMF/WSMO.

7.2.2.1. WSML & WSMX

Ganz im Sinne des Ansatzes von jABC/jETI als ganzheitliche Umgebung zum Umgang mit sämtlichen Aspekten von Remote Services existiert mit WSMX eine Referenzimplementierung der durch das WSMF bzw. durch WSMO definierten Konzepte (s.o.). WSMX bietet dabei von der Suche über die Auswahl bis hin zur Mediation und Ausführung von Semantic Web Services das komplette Spektrum an Funktionen, welches man von einem komplexen, ganzheitlichen Framework erwartet. Im Gegensatz zur modularen, dynamischen Philosophie des jABC, bzw. jETI, welches auf verschiedene Plugins zur Abbildung atomarer Funktionalitäten zurückgreift (Ausführung, Discovery, Monitoring, Prozesssynthese, etc.) handelt es sich bei WSMX um eine komplexe, statische Implementierung. Diese schließt den einfachen Austausch oder die Erweiterung der Plattform hinsichtlich neuer Technologien und Methoden durch Domänenexperten ohne tiefergehenden Einblick in die Plattform trotz des hohen Grades an Entkopplung der Einzelteile aus, erhöht jedoch das Zusammenspiel und die Kompatibilität der funktionalen Komponenten.

Wissensrepräsentation: Auf Basis von Nachrichtenanalyse und internen Anforderungen der zu verwendenden Dienste an die Datenstruktur werden zunächst Ontologien in WSML erzeugt. Die durch das Meta-Modell von WSMO definierte WSML repräsentiert eine Familie von Ontologiesprachen und bildet den von WSMO aufgespannten Sprachraum in unterschiedlichen Ausdrucksstärken ab. Sie bietet somit die formale syntaktische und semantische Grundlage der Implementierung, um Ontologien zu spezifizieren. WSML basiert auf verschiedenen logischen Formalismen, wie z.B. der Beschreibungslogik (DL) [BCM+03], der Prädikatenlogik erster Stufe (FOL) [Bar77] oder der logischen Programmierung (LP) [Llo87]. Die Varianten reichen dabei von WSML-Core, über WSML-DL bis hin zu WSML-Flight und WSML-Rule, was einer gesteigerten Ausdrucksstärke vom Schnittpunkt der Beschreibungs- und Horn-Logik [BCM+03, Hor51] hin zur komplexen logischen Programmierung entspricht. Die vollständige Sprachvariante wird durch WSML-Full definiert. WSML besitzt in allen Ausprägungen verschiedene syntaktische Repräsentationen, welche sich an einer für Menschen zugänglichen Variante, als auch an technischen Formaten wie XML oder RDF zum Austausch von Informationen zwischen verschiedenen Applikationen über das Internet orientieren. WSML kann als auf die Konzepte des WSMF ausgerichtete Beschreibungssprache zur Wissensrepräsentation und somit als Konkurrenz zur bspw. *Web Ontology Language* (OWL) [owl04] angesehen werden, bietet aber insbesondere in diesem Kontext die Möglichkeit, Ontologien aus beiden Beschreibungsräumen über eine Teilmenge der Sprachen zu verknüpfen. Die Auswertung von durch WSML definierten Ontologien geschieht über unterschiedlich komplexe Reasoner, welche den entsprechenden Ausdrucksstärken der Sprachvarianten angepasst sind. Zur Unterstützung des Benutzers bei der Erstellung von Ontologien mittels WSML existieren grafische Werkzeuge, wie z.B. der WSML Editor[10].
Im Kontext des jABC werden die durch jETI bereitgestellten Remote Service Komponenten durch eine Kombination von Datenflussinformationen auf den Ein- und Ausgabetypen beschrieben. Zugehörige Taxonomien über die verwendeten Typen und Aktionen der jeweiligen Dienste erlauben eine Abstraktion. Syntaktisch werden die Annotationen im XML Format der Service Independent Building Blocks hinterlegt. Taxonomien sind durch entsprechende Service Logic Graphen repräsentiert und liegen somit ebenfalls im XML Format vor, was eine einfache, grafische und konsistente Modellierung innerhalb des jABC ermöglicht. Service Anfragen bzgl. bestimmter Eigenschaften eines Dienstes werden je nach den verwendeten Synthesekomponenten i.d.R. über temporallogische Formeln spezifiziert und entsprechend ausgewertet.
Den flexibleren Ansatz bietet hier das WSMF, bzgl. der je nach verwendeter Ausprägung von WSML mehr oder weniger mächtigen Beschreibungssprache. Im Kontext automatisierter Komposition und Orchestrierung von Web Services hat sich, insbesondere im Zuge der SWSC, jedoch herausgestellt, dass die von

[10] `http://www.wsmo.org/TR/d9/d9.2/v0.1/20050321/`

jABC/jETI verwendeten Typ- und Aktionsinformationen durchweg ausreichen, um die gestellten Anforderungen zu erfüllen. Der Vorteil liegt demnach weniger in der Komplexität und Flexibilität von WSML, als vielmehr auf Seiten der einfachen und gerade für Domänenexperten interessanten, hinsichtlich der Modellierung konsistenten Abbildung von Service relevanten Informationen mit dem jABC.

Prozessorchestrierung und -choreographie: Prozesse in WSMX werden explizit mit Hilfe durch die Ontologien angereicherter Abstract State Machines (ASM) abgebildet, welche jeweils entweder die Orchestrierung oder die Choreographie der aufgerufenen Web Services beschreiben. Orchestrierung und Choreographie sind somit entkoppelt. Innerhalb von jABC/jETI werden die Prozesse als SLG orchestriert. Die Annotation von zusätzlichem Wissen in Form semantischer Informationen und die Choreographie der Dienste leitet sich von der jeweils verwendeten Synthesetechnologie ab (vgl. Abschnitt 5.2.2). Hervorzuheben ist an dieser Stelle besonders die einfache und konsistente Art und Weise (als SLG mit annotierten SIBs), in der die semantischen Informationen im Kontext von jETI Verwendung finden. WSMX bietet an dieser Stelle keinerlei Modularität und bleibt durch die Kombination aus WSML und ASM auf einer eher technischen Ebene.

Datenmediation: Die Mediation der Daten wird über mit Hilfe des *Web Service Modeling Toolkit* (WSMT)[11] erstellten, durch Data-Mapping Regeln ausgedrückten, objektorientierten Mediatoren durchgeführt, indem diese auf die ausgetauschten Nachrichten zwischen den Diensten angewendet werden. Zusätzlich existieren bidirektionale XML-WSML Adapter für das teilautomatisierte Lifting & Loweringder Ontologien. jETI und die verwendeten Synthesekomponenten nutzen zur Datenmediation Kontrollflussinformationen bzgl. der ein- und ausgehenden Parameter eines Dienstes, welcher im jABC als SIB repräsentiert wird. Durch den automatischen Web Service Import (siehe Abschnitt 5.2.2) sind die Parameter, welche durch komplexe Java Objekte abgebildet werden, konzeptuell bedingt korrekt. Zur automatischen Synthese von Prozessen müssen jedoch u.U. Hilfsdienste vorhanden sein oder manuell erzeugt werden, die grob den Mapping Regeln von WSMX entsprechen und eine Transformation komplexer Daten in für weitere Komponenten lesbare Formate überführt (vgl. Abschnitt 5.2.2).

Dienstrepräsentation: Modellierte Prozesse und Dienste werden intern als AXIS2 oder JAX-WS basierte Web Services bereitgestellt, wobei sich eine entsprechende Ausführungskomponente von WSMX um die Kommunikation mit den Diensten kümmert. jETI setzt bei der Kommunikation mit integrierten Web

[11] `http://sourceforge.net/projects/wsmt`

Services ebenfalls auf die Standardtechnologien AXIS2 und JAX-WS, erweitert diese jedoch aus Kompatibilitätsgründen um die Anbindung von AXIS (dem Vorläufer von AXIS2) basierten Diensten. Über entsprechende Import/Export-Mechanismen werden SIBs zur Verwendung im jABC erzeugt, welche als entsprechend parametrisierte (vgl. vorherigen Paragraph zur Datenmediation) Kommunikationsschnittstellen dienen.

Prozessausführung und -monitoring: Die, wie oben beschrieben, zur Designzeit als ASM modellierten WSMO Orchestrierungen und Choreographien sind direkt ausführbar. Zur Überwachung der Ausführung in WSMX existiert zudem eine separate Java Applikation, welche den Event-Fluss der Komponenten grafisch visualisiert darstellt. Das jABC bietet mit dem Tracer Plugin (vgl. Abschnitt 5.2) eine komplexe Prozessausführungs- und Monitoring-Komponente, welche auf beliebigen Service Logic Graphen (solange sich die SIBs semantisch als ausführbar darstellen) angewendet werden kann.

7.2.2.2. WebML, WebRatio, GLUE & BPMN

Ein weiterer, auf den Konzepten des WSMF basierender Ansatz wurde ebenfalls im Kontext der SWSC implementiert und wird in [BCV+08] detailliert beschrieben sowie mit der Technologie von jABC/jETI verglichen. Als Beschreibungssprache der Web Services kommt die Web Modeling Language (WebML) [CFB+02] zum Einsatz, welche bezogen auf die Ausdrucksstärke WSML-Flight entspricht. Die für die letztendliche Realisierung einer Applikation benötigten Dienste werden mit Hilfe der Discovery Engine GLUE [VCC05] ausgewählt und anschließend manuell als ein, in BPMN ausgedrückter, Prozess modelliert. Lediglich zur Modellierung benötigte Fragmente werden in Form von sog. Skeletons automatisiert erzeugt. Unter dem Oberbegriff des *Computer Aided Software Engineering* (CASE) wird an dieser Stelle somit eher der technische Entwickler bei der Implementierung unterstützt und weniger dem Domänenexperten zugespielt. Als CASE-Werkzeug kommt die WebRatio[12] Entwicklungsumgebung zum Einsatz. Da der konkrete Vergleich mit jABC/jETI im Kontext der SWSC in der o.a. Veröffentlichung [BCV+08], welche zum Rahmenwerk dieser Dissertationsschrift gehört, abgehandelt wird, soll an dieser Stelle auf eine komplexe Wiederholung des Inhalts verzichtet werden.

[12] http://www.webratio.com

7.2.2.3. Grundlegende Problematik des Vergleichs mit dem WSMF

Der in diesem Abschnitt angestrebte Vergleich zwischen jABC/jETI und dem WSMF sowie dessen Einordnung in den Gesamtkontext der Arbeit bezieht sich auf die gesammelten Erfahrungen im Rahmen der Semantic Web Services Challenge (vgl. Abschnitt 6.3). Die Problemstellungen der SWSC wurden dabei maßgeblich von Arbeitsgruppen des STI, bzw. DERI geprägt sowie (mit)entwickelt und stellen daher konzeptionell eine typische Problemstellung des Semantic Web im Kontext der entsprechenden Forschungsarbeiten dieser Gruppen dar. Mit dem WSMF existiert eine stark abstrahierte, rein konzeptionelle Sicht auf den gegeben Problemraum, welche im Zuge der letztendlichen Referenzimplementierungen aus den Abschnitten 7.2.2.1 und 7.2.2.2 sehr konkret gefasst werden, um die spezifizierten Anforderungen zu bewältigen. Eine Übertragung auf andere, weit allgemeinere Szenarien, insb. außerhalb der Grenzen des Semantic Web, ist daher nur mit komplexen Änderungen und Workarounds an den thematisierten Technologien zu bewerkstelligen. Hervorgehoben sei an dieser Stelle insbesondere die explizite Orchestrierung und Datenmediation auf Code-Ebene. Die Kombination von jABC und jETI bietet hier den deutlich agileren und allgemeineren Ansatz, da die beiden Frameworks nicht ausschließlich auf den konkreten Einsatz im Rahmen semantischer Web Service Anwendungen hin entwickelt wurden. Bemerkenswert ist die dabei insbesondere im Rahmen der SWSC deutlich gewordene, hohe Flexibilität des Ansatzes, welche eine große Abdeckung der definierten Problemstellungen bietet, ohne dabei auf diese beschränkt zu sein.

7.3. Service Discovery

Insbesondere die in Abschnitt 7.2 verglichenen Technologien zur automatischen Orchestrierung semantisch aufbereiteter Web Services benötigen komplexe Mechanismen zur Suche nach relevanten Komponenten, die über einfache Verzeichnisdienste, wie z.B. UDDI oder der Registry-Service Spezifikation von ebXML [con02b], hinausgehen. Diese bieten innerhalb definierter Schranken und abgeschlossener Systeme ausreichende Möglichkeiten, um entsprechende Dienste zu finden (vgl. die Suche von jETI Diensten innerhalb des jABC Repositories), stoßen jedoch an ihre Grenzen, wenn nahezu beliebige, mit semantischer Technologie angereicherte, dynamische und stark heterogene Services verarbeitet werden müssen. Dieser Abschnitt fokussiert daher den Vergleich zwischen den Discovery-Engines der Arbeiten aus Abschnitt 7.2.2.1 und 7.2.2.2 und dem im Rahmen von jETI eingesetzten miAamics Framework.

7.3.1. GLUE

Den Prinzipien von WSMO folgend stellt sich GLUE als konzeptuelles Modell zur Auffindung von Diensten dar und erweitert den durch die Web Service Modeling Ontology spezifizierten Ansatz. Insbesondere wird im Kontext von GLUE zwischen Klassen und Instanzen von Web Services und Zielbeschreibungen unterschieden, was eine differenziertere Sicht hinsichtlich abstrakter und konkreter Elemente ermöglicht. Goal-Klassen können als Vorlage zur abstrakten Anfrage von Diensteigenschaften angesehen werden, welche der Nutzer mit konkreten Werten befüllt, um seine Suche zu spezifizieren. Web Services werden in Form abstrakter Klassifikationen und konkreter Instanzen eines Dienstes repräsentiert, was der Taxonomie-basierten Betrachtung der Dienste und Eigenschaften von miAamics entspricht. Die Spezifikationen basieren dabei auf WSMO Ontologien in F-Logic. Ein weiteres charakteristisches Merkmal der GLUE-Engine ist die zentrale Rolle sog. Mediator-Komponenten zur Vermittlung zwischen Web Services und Goals (vgl. Abschnitt 7.2.2). Die Implementierung dieses, durch das WSMF konzeptuell definierten, Mediators beinhaltet Regeln zur Evaluation der Anforderungen gegen einen Dienst, um ein vorgegebenes Ziel zu erreichen und bildet damit die Kernfunktionalität des Matching-Prozesses der Discovery-Engine ab. Der komplette Prozess zur Auffindung eines relevanten Dienstes gliedert sich dabei in drei aufeinander aufbauende Phasen:

1. Zur **Setup** Zeit werden die Web Service und Goal-Klassen erzeugt und geladen.
2. In der **Publishing** Phase werden die von den Anbietern zur Verfügung gestellten Web Services auf Basis der Klassen aus Phase 1 instanziiert und veröffentlicht
3. Während der **Discovery** Phase instanziiert der Benutzer auf Basis der Goal-Klassen aus Phase 1 ein Ziel und übermittelt die Anforderungen an das System. Als Ergebnis erhält er eine nach ihrer Relevanz bzgl. der Zielspezifikation geordnete Liste von Diensten, die das Ziel vollständig und/oder partiell erfüllen.

Der Suchprozess der GLUE Discovery Umgebung basiert dabei auf der über F-Logic arbeitenden Flora-2 [YK01] Engine, einem Tableau-basierten Prolog Reasoner, der die reine Matching-Funktionalität bereitstellt. GLUE gliedert die prinzipbedingt in F-Logic spezifizierten Konstrukte der Web Service und Goal-Beschreibungen sowie Ontologien und Mediatoren über Wrapper Konstrukte in das WSMO Konzept ein. Das Framework selbst stellt eine Web Service Schnittstelle dar, was es für externe Frameworks leicht zugänglich und integrierbar macht. Die Implementierung der Mediatoren zwischen Web Services und den Zielen erlauben es im Kontext von GLUE verschiedene Einstiegsregeln zu defi-

nieren, welche über entsprechende Prioritäten einer entsprechenden Erfolgsstufe des Matching-Prozesses zugeordnet werden. Mit ihrer Hilfe können somit diskrete Abstufungen in der Ergebnismenge erzielt werden, was zu exakten bis hin zu partiellen Lösungsmengen für eine konkrete Discovery-Anfrage führt. Mit Hilfe des sog. Data-Fetching (Aufruf von sicheren Methoden ohne Randbedingung nach WSDL 2.0) ist es möglich, innerhalb von GLUE dynamische Parameter, bspw. arithmetische Funktionen, zu verwenden. Während des Matching-Prozesses ist die Engine somit in der Lage, notwendige Werte dynamisch abzufragen oder zu berechnen und auch auf kritische Variablenwerte von Diensten zurückzugreifen.

7.3.2. WSMOLX

Basierend auf den Konzepten des WSMF stellt WSMOLX die Referenzimplementierung einer Discovery Engine in diesem Kontext dar. Dienste werden in Form von WSMO Web Services durch ihrer Funktionalitäten (Capabilities) und die Schnittstellen (Interfaces) in WSML-Flight beschrieben (vgl. Abschnitt 7.2.2.1). Die Ziele einer Suchanfrage sind analog dazu durch die geforderten Möglichkeiten hinsichtlich Funktionalität und Interaktion als Nachbedingungen in Form boole'scher Regeln und arithmetischer Ausdrücke spezifiziert. Auf Basis der modellierten Ontologie wird letztendlich mit Hilfe des IRIS Reasoner[13] eine Lösung gesucht. Die Ergebnismenge wird dabei durch im Ziel der Anfrage spezifizierte Bewertungseigenschaften sortiert und qualifiziert ausgewählt, so dass eine Rangfolge der besten Ergebnisse ausgegeben werden kann. Der detaillierte Vergleich zwischen jETI/jABC in Kombination mit miAamics und WSMOLX ist der für diese Dissertation relevanten Veröffentlichung [KVW+08] zu entnehmen, weshalb an dieser Stelle nicht weiter darauf eingegangen werden soll.

7.4. Web-basierte Ansätze

In diesem Abschnitt sollen Web-basierte (Portal-)Konzepte zur Ausführung und Bereitstellung von Diensten im Kontext von serviceorientierten Architekturen vorgestellt und diskutiert werden. Vergleichend wird hier insbesondere auf die geplante jETI-Testumgebung (*jETI-Testbed*, siehe Abschnitt 9) zur Evaluation und einfachen Handhabung von Diensten Bezug genommen.

[13] `http://iris-reasoner.org`

7.4.1. REportal

Die im Jahre 2001 von Mancoridis et al vorgestellte Portallösung *REportal* [MSC+01] auf Basis einer dynamischen Website ist im Kern eine Sammlung von Diensten zum Reverse Engineering von Applikationen in diversen Programmiersprachen. Dem Grundkonzept von „Software as a Service“ (SAAS) folgend besitzt REportal den Anspruch einer einfachen, verlässlichen und leicht zugänglichen Benutzung von komplexen Softwarewerkzeugen ohne zeitaufwändige Installation, komplizierte Konfigurationsarbeiten oder Versionspflege durch den Benutzer, und skizziert dabei genau jene Motivation, die ursprünglich Mitte der 90er Jahre zur Entwicklung und Integration des ersten ETI Prototypen im Rahmen der damaligen ABC Modellierungsumgebung führte (vgl. Abschnitt 5).
Konzeptuell lässt sich REportal in den Kontext des Web-basierten Software Engineering einordnen. Die Autoren sehen Ihre Arbeit selbst im Zusammenhang mit anderen Entwicklerplattformen wie z.B. *SourceForge*[14] oder *Software Bookshelf* [FH+97]. Dementsprechend dient das Portal der einfachen Benutzung einzelner Services durch den Anwender, indem Eingaben bereitgestellt und Ausgaben im Anschluss präsentiert werden. Der Nutzer wählt dabei durchgängig durch Wizards unterstützt aus einer Reihe von Services einen bestimmten Dienst aus und stellt diesem via Upload seine Eingabedaten zur Verfügung. Die Mediation der Daten in beide Richtungen und die Ausführung des Dienstes übernimmt ein entsprechendes Servlet unter Berücksichtigung von Sicherheitsmechanismen zur Authentifizierung bzgl. Ausführung und Dateizugriffen sowie Datenbankanfragen. Die Architektur von REportal stellt sich somit als dreischichtiges Modell dar (User Interface, Servlet, Services). Besonderes Augenmerk wurde auf die Erweiterbarkeit des Portals gelegt, so dass neue Dienste relativ einfach in diese Architektur integriert werden können, indem sie auf unterster Ebene auf einem zentralen Toolserver hinterlegt und über das Servlet und letztendlich das User Interface eingebunden werden. Obwohl zum Zeitpunkt der Veröffentlichung auf die Domäne des Reverse Engineering von Software beschränkt, ist es somit prinzipiell denkbar, die Plattform zu öffnen und domänenübergreifende Werkzeuge wie z.B. Tools zur statistischen Auswertung von Daten, etc. bereitzustellen (vgl. Abschnitt 6.1).
Eine komplexe Orchestrierung der einzelnen Dienste ist nicht vorgesehen. Werden jedoch zur Ergebnispräsentation einer Ausgabe bestimmte Zusatzdienste benötigt, so werden diese als Tool-Kette im Sinne eines Filters hinter den ursprünglich vom Benutzer angefragten Service geschaltet. Eine Ausführung der Dienste über entsprechende Remote Schnittstellen und somit eine externe Nutzung im Rahmen komplexer Modellierungswerkzeuge ist ebenfalls nicht möglich, da es dem von Mancoridis et al in diesem Zusammenhang propagierten Ansatz der einfachen Benutzung der Tools über das von REportal bereitgestellte User Interface

[14] http://sourceforge.net

(Website) widerspräche und das Konzept aufweichen würde.
Die Stärken von REportal liegen ganz klar in der sofortigen konfigurations- und installationsfreien Benutzung der angebotenen Dienste, über ein Web Interface. Benutzer unterschiedlicher Domänen sind es heutzutage gewöhnt mit Webseiten zu interagieren, so dass die technische Hürde hier auf ein Minimum reduziert wird. Ein ähnliches Konzept, Services als Sammlungen zum einfachen ausprobieren über ein Web Portal bereitzustellen, liegt mit dem geplanten Testbed für jETI ebenfalls vor (vgl. Abschnitt 9), ist aber bisher nicht praktisch umgesetzt worden. Hier liegt der Schwerpunkt allerdings weniger auf einer produktiven Benutzung der Dienste über eine Website, als vielmehr auf der Evaluation einzelner Tools, um sie im Anschluss im Rahmen komplexer Orchestrierungen mit Hilfe des jABC zu verwenden. Hier gibt es aktuell ebenfalls Bestrebungen, einen Online-Client zu implementieren, mit dessen Hilfe der Nutzer unter Verwendung diverser (Remote) Services komplexe Modelle komplett im Web erstellen kann und somit eine „echte“ Service-Orchestrierung bieten wird. Diese Möglichkeit besteht mit REportal nicht. Ebenfalls ist es nicht vorgesehen, die von REportal bereitgestellten Dienste in externen Modellierungsumgebungen bereitzustellen, da die Dienstintegration manuell auf Seiten des Servers mit Hilfe eines Servlets und der entsprechenden Repräsentation im User Interface stattfinden muss. Aus der zum Vergleich herangezogenen Arbeit geht darüberhinaus nicht hervor, ob hier standardisierte Schnittstellen zum Einsatz kommen. Dem entgegen stehen die bei der Implementierung von jETI stets im Blickfeld behaltenen offenen Standards zur Anbindung der Dienste innerhalb des jABC. jABC/jETI bietet zudem die Möglichkeit, orchestrierte Modelle automatisiert als Remote Service zur Verfügung zu stellen und vereinfacht somit nicht nur die Nutzung der angebotenen Dienste, sondern zugleich die Bereitstellung durch den Service Provider, der ebenfalls die Vorteile einer strikt serviceorientierten Architektur nutzen kann. Ebenso können durch die Verwendung von Remote Diensten innerhalb dieser modellierten und wiederum als Remote Service angebotenen Komponenten Hierarchien von verteilten Dienstausführungen gebildet werden. Diese laufen im Rahmen der jETI Remote Service Umgebung dezentral auf verschiedenen Service Provider Sites ab, denen ein von REportal genutzter zentraler Toolserver gegenübersteht. Dienstanbieter können somit ihre Repositories selbst verwalten und/oder redundant zur Verfügung stellen, was die Performance bei einer Vielzahl gleichzeitiger Tool-Ausführungen sowie die Ausfallsicherheit signifikant erhöht.

7.5. Service Mashups

Der aufkeimende Begriff der *Service Mashups* beschreibt in populistischer Form nichts anderes als die Orchestrierung von Diensten verschiedener Anbieter zu einem funktional neuen Gesamtkonstrukt. Konzeptuell wird also keine Neuerung

im Vergleich zur klassischen Service Orchestrierung geschaffen. Mashups zeichnen sich jedoch häufig durch die Tatsache aus, dass eine hohe Anzahl heterogener Dienste, üblicherweise SOAP basierte Web Services, verknüpft wird und die angebotenen Frameworks eher einfach gehalten sind. Oftmals werden die Konzepte zudem auf der technischen Ebene von Programmiersprachen umgesetzt und eher selten ganzheitliche Umgebungen zur grafischen Modellierung geschaffen. Aus den vorangestellten Gründen wird diese Gruppe von Frameworks und Ansätzen in diesem Abschnitt gesondert betrachtet. Mashups sind prinzipbedingt mit jeder der in den vorherigen Teilen dieser Arbeit beschriebenen Modellierungsumgebungen und jedem der konzeptuellen Ansätze realisierbar, was einen direkten Vergleich im Grunde unmöglich macht, da u.a. unterschiedliche Zielgruppen adressiert werden. Dennoch soll der Vollständigkeit halber im folgenden eine Auswahl von Mashup Konzepten im konzeptuellen Vergleich zu jETI beschrieben werden.
Es existieren eine ganze Reihe unterschiedlicher Ansätze, um den Umgang mit (Web) Service Mashups zu vereinfachen. Z.B. bietet Google mit seinem auf AJAX [Gar05] basierenden Web Toolkit (GWT) eine Möglichkeit mit relativ geringem Aufwand verschiedene Dienste zu kombinieren, die Google wiederum selbst bereitstellt. Ein weiteres populäres Framework ist durch *Ruby on Rails*[15] gegeben, welches auf der objektorientierten Programmiersprache *Ruby*[16] basiert. Sieht man Java als die Grundlage für jABC/jETI an, so bildet Ruby analog die Basistechnologie für das Rails Framework. Eines der Basiskonzepte von Ruby ist hierbei die Minimierung der Codezeilen, die ein Softwareentwickler produzieren muss, um zu einer voll funktionsfähigen Applikation zu gelangen. Das Rails Framework erweitert dieses Paradigma hinsichtlich Web-basierter Service Mashups indem es sich strikt an den pragmatischen *Don't Repeat Yourself* (DRY) [HTC99] und *Convention over Configuration* (COC) Konzepten orientiert und den Benutzer so in agiler Softwareentwicklung unterstützt. Im Vergleich zu der in dieser Arbeit entwickelten Remote Service Umgebung und insbesondere unter dem Aspekt der Semantik von Modellen im jABC ist ein deutlich ähnlicheres Framework durch ORC [Mis04] gegeben. Benutzer konzentrieren sich dabei auf das Datenflussmodell der Applikation und können existierende Dienste orchestrieren. Die Kompositionen werden durch eine minimale Programmiersprache realisiert, welche durch die Kleen'sche Algebra [Con71] inspirierte Basisoperatoren zur Verfügung stellt. Generell ist bei nahezu allen zum Zeitpunkt dieser Arbeit verfügbaren Technologien eine Gemeinsamkeit festzustellen: Der Benutzer muss mehr oder weniger fundierte Kenntnisse im Umgang mit Programmiersprachen und/oder den technischen Aspekten der Softwareentwicklung mit sich bringen, um die entsprechenden Werkzeuge konzeptuell erfassen und in vollem Umfang nutzen zu können. Verglichen mit dem jABC/jETI Ansatz bieten Ruby on Rails, das GWT, AJAX oder die einfache Nutzung einer beliebigen Programmiersprache in Verbindung mit einem Web

[15] `http://rubyonrails.org/`
[16] `http://www.ruby-lang.org/`

Service Framework wie z.B. AXIS oder JAX-WS (welche auch Basisfunktionalitäten in vielen ausgewachsenen Frameworks bereitstellen, vgl. Abschnitt 7.2.2) dem Entwickler deutlich mehr Flexibilität und technische Tiefe. Diese Aussage lässt sich allerdings relativieren, wenn man bedenkt, dass ein Nutzer mit entsprechend technischem Vorwissen durchaus in der Lage ist, seine eigenen Komponenten (SIBs) innerhalb von jABC/jETI zu entwerfen und zu implementieren. Im Gegensatz zu den vorher genannten Technologien ist dieses Wissen dabei aber nicht grundsätzlich notwendig, um mit der Remote Service Umgebung komplexe Serviceorchestrierungen vorzunehmen. In diversen Fallstudien (siehe Abschnitt 6) konnte nachgewiesen werden, dass Nutzergruppen, welche mit heutigen Standards auf dem Gebiet der Programmierung und Serviceorientierung keinerlei Erfahrung mitbrachten, durchaus in der Lage waren, innerhalb kurzer Zeit mit dem System zu arbeiten und zu umfangreichen Ergebnissen zu gelangen. Zwischen den beiden Extremen der fundamentalen Programmierung und der grafischen, werkzeuggestützten Modellierung ist die ORC Programmiersprache anzusiedeln, da sie einerseits durchaus der Philosophie der Serviceorchestrierung auf Basis von Datenflussstrukturen folgt, andererseits dem Benutzer jedoch zumindest rudimentäre Kenntnisse allgemeiner Programmierkonzepte abverlangt.

7.6. Zusammenfassung

Abschließend soll an dieser Stelle eine strukturierte Zusammenfassung der in den vorherigen Abschnitten betrachteten Arbeiten gegeben werden. Die konkreten Ansätze aus Abschnitt 7.1 zur modellgetriebenen Entwicklung und Abschnitt 7.2 zu Semantic Web Services und Syntheseansätzen werden in Tabelle 7.6 gegenübergestellt. Die konzeptuellen Ansätze und Arbeiten, die nur einen geringen Teil der Vergleichskriterien abdecken, werden dabei nicht betrachtet. Um den Teilbereich der Discovery und somit die Arbeiten aus Abschnitt 7.3 ebenfalls adäquat vergleichen und gegenüberstellen zu können, sind diese Ansätze separat in Tabelle 7.6 aufgeführt.

Funktion	**Seaside**	**Taverna**	**Service Composer**	**WSML/ WSMX**	**WebML/ WebRatio**	**jABC/jETI**
Prozessor-chestrierung	Kontroll-flussabbil-dung in Smalltalk	Abstrakter SCUFL Workflow	BPEL4WS Prozessmodell	Abstract State Machines	BPMN Prozessmodell	Abstrakter Service Logic Graph mit wechselnder Semantik, je nach Anwen-dungsfall und verwendeten Erweiterungen

Tabelle 7.1.: Übersicht der verwandten Arbeiten (Frameworks & Semantic Web)

Funktion	Seaside	Taverna	Service Composer	WSML/ WSMX	WebML/ WebRatio	jABC/jETI
Dienstre-präsentation	Atomare Blöcke von Source Code Fragmenten zur Kapselung minimaler Applikations-funktionalitä-ten	XML-Prozessoren mit konkreten Funktionsauf-rufen	SOAP basier-te Web Ser-vices	Repräsentative Web Ser-vice Objekte (Choreographie-Komponenten)	Parame-trisierbare WebML standard Einheiten (Units)	Parame-trisierbare SIBs zur Kapselung der Dienstimple-mentierung
Dienstinte-gration	API Anbin-dung von Web 2.0 Technologien zur Unter-stützung von Web Services	Pluginkon-zept zur statischen Anbindung frei/manuell implemen-tierten und registrierten Web Service Bibliotheken sowie REST API	Integration beliebiger SOAP basierter Web-Services über ent-sprechende Frameworks	Automatische Generierung von Skeletons und Stubs über AXIS2 oder JAX-WS Framework	Automatische Generierung von Skeletons und Stubs über ent-sprechende WS-Frameworks	Automatische Integration von jETi Diensten, Web Services (über entspr. Frameworks), REST, CORBA, etc.. Erweiterbar über API Anbindung
Wissensre-präsentation	Keine integrierte Techno-logie zur Wissensre-präsentation vorhanden	RDF Onto-logien und semantisch annotierte Web Services mit Hilfe des SADI Frameworks	Einfache Methoden-signaturen der Dienste und OWL-S oder WSMO Ontologien	WSMO Ontologien in WSML	Aus BPMN Modell abgeleitetes WebML ER-Modell mit automatisch generierten Standard-Komponenten	Taxonomien als SLG und semantische Annotationen von DFA Informationen an den SIBs (Kontroll-flussinforma-tionen bzgl. der Ein- und Ausgabepa-rameter eines Dienstes, welche mit Hilfe von Transformati-onsdiensten zugeordnet werden)
Mediation	Manuelle Konfiguration der Code-Fragmente	Manuelle Konfigu-ration der Prozessoren	Dienstfunk-tionalitäten werden über eine Anfra-gesprache formuliert. Komplexe Nebenbedin-gungen führen zu einer Interaktion mit dem Benutzer, was nur eine teil-automatische Mediation ermöglicht. Partielle Tref-fer können berücksichtigt werden	Durch Data-Mapping Regeln mit dem WSMT spezifizierte Mediatoren werden auf die Nachrich-ten zwischen den Diensten angewendet	Manuelle Konfiguration der generierte Standard-Komponenten des WebML Modells	Manuelle Or-chestrierung als SLG oder automatische Prozesssyn-these auf Basis der DFA Anno-tationen und Taxonomien
Ausführung	Ausführungs-und Debugging-Komponente mit Zustand-sabbildung und Spei-cherung der Kontrollfluss-ausführung	Komponenten zur Ausfüh-rung der Prozessoren im System, von der Komman-dozeile oder als Remote-Workflow	Ausfüh-rung der BPEL4WS Modelle durch geeignete Business Process Engine als Interpreter	WSMX Ausführungs-komponente	WSMX Ausführungs-komponente	Zustands-basierte Ausführungs-umgebung auf dem Kontrollfluss als Plugin-Komponente

Tabelle 7.1.: Übersicht der verwandten Arbeiten (Frameworks & Semantic Web)

Funktion	Seaside	Taverna	Service Composer	WSML/ WSMX	WebML/ WebRatio	jABC/jETI
Monitoring	Fehler- und Zustands-anzeige als Webseite mit Bezug zur Entwicklungs-umgebung über die Debugging-Komponente der Ausführ-ungsumge-bung	Workflow Monitor Komponente als REST Web Service zur periodischen Aktivitäts-überwachung unterstützter Dienst-Pakete (WSDL, SoapLab, Biomoby und Biomart). Historische und aktuelle Zustandsda-ten werden ausgewer-tet und visualisiert.	Monitoring der Prozess-ausführungen durch die Business Process Engine, falls diese entsprechende Möglichkeiten anbietet.	Java Appli-kation zur grafischen Aufbereitung des Event Flusses	Komponente zur Generie-rung von Web Seiten mit Datenbankin-formationen	Überwachung der Dienstzu-stände und -parameter über die Ausführungs-komponente
Discovery	Manuelles Auffinden von Diensten notwendig	Plugin-Schnittstelle zur Integrati-on beliebiger Discovery Kompo-nenten, zugeschnitten auf die jeweiligen Dienst-Pakete	Verzeich-nisdienst (UDDI)	WSMX Discovery Framework (Schlüssel-wörter, DL, regelbasiert)	GLUE Engine	Verzeichnis-dienst (SIB Bibliothek), miAamics Engine
Zieltransfor-mation und Dienstex-port	XHTML	Web Service	Web Service	Web Service	Web Services als vollwerti-ge J2EE Ap-plikation	Mit Hilfe der Code-Generator Plugin-Komponente beliebige Zieltransfor-mationen als Applikation und/oder (Web) Service möglich
Prozesssyn-these	keine Synthese-komponenten integriert	keine Synthese-komponenten integriert	AI-Planning und Optimie-rungsalgorith-men	Regelbasierte Choreogra-phie der Dienste auf Basis von Reasoner-Auswertungen auf den Ontologien (angepasst an die verwendete Model-lierungs-sprache). Manuelle Konfiguratio-nen im Code der Modelle notwendig	Manuelle Konfiguration des WebML Modells notwendig, welches automatisch aus dem BPMN Mo-dell abgeleitet wird	Über eine standardi-sierte API können un-terschiedliche Planning- und Synthese-Komponenten genutzt werden

Tabelle 7.1.: Übersicht der verwandten Arbeiten (Frameworks & Semantic Web)

Funktion	GLUE	WSMOLX	jABC/miAamics
Dienstbeschreibung	Abstrakte Web Service Klassen als Vorlage in F-Logic, welche durch den Einsatz konkreter Werte instanziiert werden	Vor- und Nachbedingungen eines Dienstes, Schnittstelle zur Ausführung und Datenrepräsentation, Regeln mit Rückgabe von numerischen Werten, arithmetische und zusätzlich definierte Operatoren auf den Dienstbeschreibungen	SLGs mit SIBs als atomaren Diensten und Parametern, arithmetische Auswertungen erfordern Vorberechnung im SLG, gewichtete Regeln, berechnete Auswertungsstruktur zur Ergebnisfindung
Zielbeschreibung	Abstrakte Zielklassen analog zu den Dienstbeschreibungen mit gewichteten Regeln in F-Logic	Nachbedingungen (harte Bedingungen), Referenzierungen zu boole'schen und arithmetischen Regeln, nicht funktionale Eigenschaften	Priorisierte Boole'sche Regeln und Strategien (Kombinationen von Regeln)
Datenmodell	WSMO Ontologien in F-Logic	WSMO Ontologien	Taxonomien aus Attributen und Kategorien (Kombination von Attributen) auf Basis der Parameter eines SIBs
Ergebnis-Matching	Regelauswertung auf den Ontologien mit Flora-2 Reasoner basierend auf Service- und Zielinstanzen, dynamische Parameter mit Datenanfragen während der Auswertung	Regelauswertung mit IRIS Engine (Datalog Reasoner, logische Programmierung) auf den Ontologien, dynamische Berechnung auf Basis von Variablen der Diensteigenschaften	Regelauswertung auf Basis von ADD auf der abstrakten Menge der Taxonomien (entkoppelt von konkreten Diensten), Vorgenerierung einer Auswertungsstruktur
Ergebnisselektion	Ranking auf abstrakten Web Service Klassen	Ranking auf den nicht funktionalen Eigenschaften einen Dienstes, konkrete Ergebnismenge von Diensten	Auswahl der besten Treffer aus der geordneten Ergebnismenge basierend auf den Gewichtungen der einzelnen Regeln, abstrakte Ergebnismenge auf der Taxonomie

Tabelle 7.2.: Übersicht der verwandten Arbeiten (Discovery)

Teil III.

Fazit und Ausblick

8. Fazit

In dieser Arbeit wurde das Konzept der jETI Remote Service Umgebung zur intuitiven Orchestrierung von verteilten heterogenen Diensten unterschiedlicher Domänen auf Basis grafischer Modelle vorgestellt. Heutige Frameworks zur möglichst umfangreichen Behandlung aller Aspekte von verteilten Diensten stellen i.d.R. eine Teilmenge des von jETI bereitgestellten ganzheitlichen Ansatzes dar und spezialisieren sich dabei häufig auf einzelne Domänen und konkrete Technologien. Des Weiteren ist zu beobachten, dass zwar durch den Anspruch der möglichst umfangreichen Umsetzung technologischer Konzepte, komplexe und flexible Lösung entstehen, diese aber allzu häufig ein hohes Maß an Vorwissen bzgl. der eingesetzten Methoden und Formalismen voraussetzen. Häufig wird dabei ein Konzept zur Code-basierten Integration und Kapselung unterschiedlicher Dienste bereitgestellt jedoch speziell die vereinfachte Integration von Remote Services außer Acht gelassen, da diese i.d.R. als „normale“ Dienste aufgefasst werden und sich nur durch den implementierten Code im Rahmen des Service-Grounding als Client-Stub, bzw. Kommunikationsschnittstelle des Remote Dienstes von diesen unterscheiden. Das Integrationsproblem wird somit auf die Ebene des Komponentenentwicklers verlagert und speziell Anwendungs- und Domänenexperten ohne tiefgreifendes Verständnis der einzusetzenden Remote-Technologien stehen dabei vor der Aufgabe, solcherlei Dienste autonom, einfach, effizient und korrekt in eine bestehende Umgebung einzubinden.
Trotz des durchaus betagten ursprünglichen Kernkonzepts, eine eigene proprietäre und möglichst abstrakte Integration von verteilten Diensten zu realisieren, ist die Idee von ETI als Vorläufer der hier präsentierten Arbeit daher nach wie vor aktuell. Bis heute dauern intensive Forschungsarbeiten auf diesem Gebiet an und es werden immer neue technologische Ansätze präsentiert, welche zu einer starken Fragmentierung des Segments verteilter Dienstintegration beitragen. Durch die Vereinigung möglichst vieler dieser als sinnvoll erachteten und industriell akzeptierten Methoden sowie dem stark abstrahierenden Grundansatz von ETI gepaart mit der strengen Anlehnung an die Konzepte des XMDD, wurde somit insbesondere in Kombination mit der jABC Modellierungsumgebung eine einzigartige Umgebung geschaffen, welche nicht auf einzelne Technologien beschränkt ist und durch ihre offene Struktur auch in Zukunft Erweiterungen hinsichtlich neuer Erkenntnisse in diesem Bereich ermöglicht. Das erklärte Ziel der Arbeit war dabei, die drei Hauptachsen der individuellen Modellierung von Domänen (Domain Modeling), abstrakter, entkoppelter und „loser“ Programmierung im Sinne

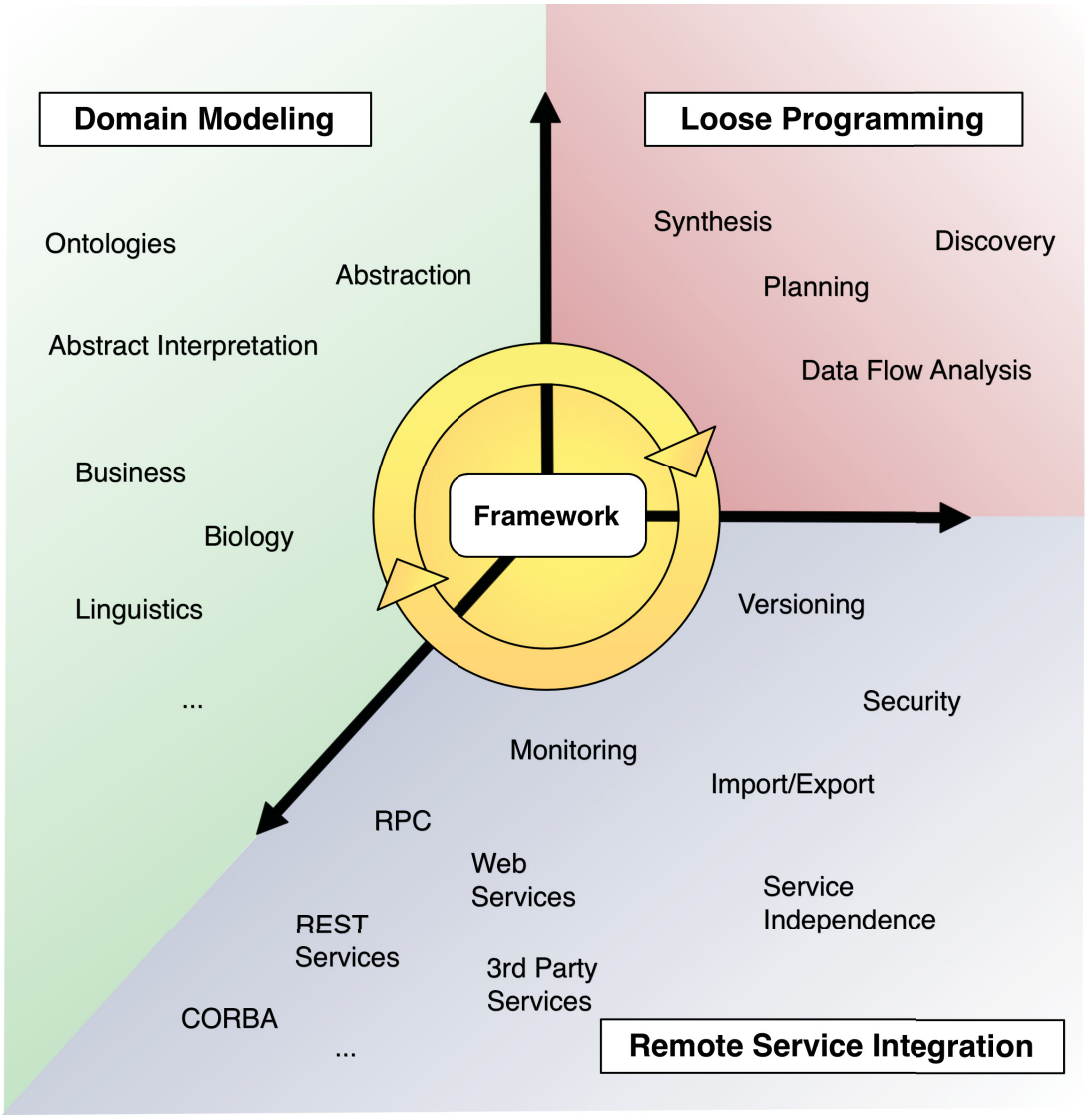

Abbildung 8.1.: Die ganzheitliche Sicht des Modellierungs-Framework

von automatisierten Prozessmodellen (Loose Programming) und der Integration von verteilten Diensten (Remote Service Integration) ganzheitlich in einem Framework zu vereinen und zu gewährleisten. Abbildung 8.1 veranschaulicht die abstrakte und ganzheitliche Sicht auf die Umgebung im Sinne des konzeptuellen Ziels.

8.1. Anforderungsanalyse

Um die einzelnen Aspekte des Konzepts zu demonstrieren und zu evaluieren wurde eine ständig erweiterte Referenzimplementierung geschaffen, welche in unterschiedlichen Kontexten und Domänen bestehen konnte. Eine bestmögliche Betrachtung der Vorteile und des Potentials von jETI ergibt sich dabei durch die abschließende Auswertung der in Abschnitt 1.1 formulierten Anforderungen, welche im folgenden rückblickend veranschaulicht und durch die in Abschnitt 6 präsentierten Fallstudien praktisch belegt werden.

A1 - Modularität/Kompositionalität

Die Komposition und Orchestrierung von verteilten Diensten wird in der jABC Modellierungsumgebung in Form von Service Logic Graphen durchgeführt. Die integrierten Remote-Dienste werden innerhalb des Graphen durch standardisierte Service Independent Building Blocks repräsentiert, welche in einem Dienstverzeichnis vorgehalten werden. Im Kontext der jETI Remote Service Umgebung beinhalten die SIB-Komponenten im Wesentlichen die Kommunikationsschnittstelle zum Aufruf und Datenaustausch mit dem jeweils angebunden Dienst. Komplexe Orchestrierungen können zudem hierarchisch gekapselt als eigenständige Komponenten bereitgestellt werden. Durch den einfachen Austausch der SIBs zur Modellierungszeit ergibt sich somit ein hoher Grad an Modularität betreffend komponierter und atomarer Dienstaufrufe. Hinsichtlich der Kompositionalität von Diensten stehen dem jABC verschiedene Plugins zur lokalen und globalen Verifikation von Diensteigenschaften (Local Checker, Model Checker) zur Verfügung.

A2 - Universalität

Um ein möglichst universell einsetzbares Framework zu schaffen, werden etablierte Standardtechnologien zur Integration SOAP und REST basierter Web Services und RPC Aufrufe auf der Ebene des Programmcodes der SIBs von jETI direkt adressiert. Darüberhinaus existiert ein durch jETI definiertes und standardisiertes Interface zur Anbindung beliebiger Remote-Technologien und deren Kapselung innerhalb von SIBs im Rahmen des jABC. Innerhalb diverser Fallstudien konnte der universelle Einsatz von jETI-Diensten im Kontext vielfältiger und unterschiedlicher Domänen und Anwendungsbereiche evaluiert werden.

A3 - Monitoring/Benchmarking

Die Ausführung von modellierten Prozessen im jABC wird über das Tracer-Plugin realisiert, welches die Dienstimplementierung der einzelnen SIB-Komponenten zur Abarbeitung der jeweils gekapselten, atomaren Funktionalitäten heranzieht. Durch die jETI Remote Service Umgebung integrierte Dienste delegieren dabei den Aufruf des über die Kommunikationsschnittstelle innerhalb der SIBs spezifizierten Dienstes an die Ausführungskomponente. Durch ein direktes, visuelles Feedback auf dem Prozessmodell, ermöglicht der Tracer dabei das Überwachen der aktuellen Ausführung ähnlich eines Debuggers bei der Ausführung von Programmcode. Dienstzustände und Ausführungspfade können somit effizient überwacht und ausgewertet werden.

A4 - Versionierung auf Modellebene

Dem Best-of-Breed Ansatz folgend existiert keine standardisierte und vorgegebene Technologie zur Versionierung von Prozessmodellen innerhalb der jETI Remote Service Umgebung. SLGs werden im jABC in Form von XML Dokumenten persistiert und können somit dauerhaft gespeichert und exportiert werden. Über ein beliebiges Versionierungssystem, wie bspw. GIT[1], SVN[2] oder CVS[3] können die Modelle somit in unterschiedlichen Versionen und Ausprägungen dauerhaft verfügbar gemacht werden.

A5 - Plattformunabhängigkeit

Durch die prinzipiell systemunabhängige Speicherung der Service Logic Graphen als XML Dokument ist der Austausch der Modelle zwischen verschiedenen Instanzen des jABC und darüberhinaus gewährleistet. Die SIBs als atomare Bausteine der Prozesse sind dabei vollständig von der Implementierung des aufzurufenden Dienstes entkoppelt und werden erst durch das Service-Grounding, also der Verbindung der abstrakten Dienstrepräsentation mit dem gegenständlichen Service, konkretisiert. Die Auswahl und Integration der von jETI bereitgestellten Dienste kann vollständig über eine Web-basierte Schnittstelle innerhalb eines Web-Browsers und somit unabhängig von der zu Grunde liegenden Plattform durchgeführt werden. Die Standardimplementierung der SIBs sowie der Remote Service Umgebung und dem übergeordneten Modellierungs-Framework ist zudem als solche plattformunabhängig in Java implementiert, was eine maximale Inde-

[1] `http://git-scm.com/`
[2] `http://subversion.apache.org/`
[3] `http://www.cvshome.org/`

pendenz auf allen Ebenen von der Modellierung bis zur technischen Realisierung gewährleistet.

A6 - Erweiterbarkeit

Durch das modular und konsistent umgesetzte Plugin-Konzept des jABC ist eine konzeptuelle Erweiterbarkeit der Modellierungsumgebung jederzeit gegeben. jETI profitiert somit von den angebotenen Schnittstellen und kann nach Belieben über abhängige oder maximal entkoppelte Komponenten erweitert werden. Auf der Ebene der Remote Service Umgebung sind darüberhinaus standardisierte Schnittstellen zum Aufruf von verteilten Diensten und der Annotation zusätzlicher Beschreibungsmerkmale in Form semantischer Annotationen definiert, was eine Erweiterbarkeit auch bzgl. der integrierten und funktional relevanten Remote-Technologien ermöglicht.

A7 - Verifikation

Die Verifikation von Modellen wird im jABC über das GEAR-Model-Checking-Plugin [BMRS07] bereitgestellt. Temporallogische Spezifikationen, bspw. auf Basis der Computational Tree Logic (CTL) oder dem modalen μ-Kalkül [Koz83] werden validiert und die Auswertung dabei optional direkt im Modell visualisiert oder aufbereitet angezeigt (vgl. SLTL-Prozesssynthese in Abschnitt 5.2.2). Eine zusätzliche Hilfskomponente des Plugins erlaubt es zudem, die mitunter komplexen Formeln als Eingabe für den Model Checker ohne konkretes Detailwissen grafisch unterstützt zu definieren, was die Benutzung der Modellverifikation einer breiten Nutzergruppe ermöglicht.

S1 - Service Level Agreement

Zur Vereinbarung dienstrelevanter Aspekte nutzt jETI insbesondere die zu einem Dienst auf dem Komponentenserver hinterlegten Lizenz- und entsprechend korrespondierenden Anmeldeinformationen eines angebundenen Verzeichnisdienstes. Services können restriktiv aufgerufen und in ihrem Funktionsumfang, bspw. über die Anzahl der Aufrufe, eingeschränkt werden. Ebenso ist eine Abrechnung auf Basis der abgerufenen Dienstfunktionalitäten möglich, was insbesondere einem kommerziellen Einsatz der Technologie (vgl. SAAS-Konzept) zuträglich ist. Durch die generell über definierte Schnittstellen innerhalb der SIBs realisierte, entkoppelte Dienstimplementierung ist es zudem möglich, verschiedene standardisierte Sicherheitsprotokolle und Verschlüsselungen in Abhängigkeit der aufzurufenden

Dienste zu verwenden (z.B. SOAP basierte Web Services mit WS-Security Informationen oder per SSL/TLS verschlüsselte Dienstaufrufe via HTTPS).

S2 - Dienstintegration

Die intuitive und möglichst automatisierte Integration von verteilten Diensten wird innerhalb der jETI Remote Service Umgebung konzeptuell über drei unterschiedliche Ansätze realisiert. Die auf der Seite der Service Provider Site im Web-Interface oder als SLG innerhalb des jABC spezifizierten Dienstausführungen können von jETI automatisch als SIB in das Dienstverzeichnis der Modellierungsumgebung integriert werden. Web Services mit entsprechend formalen und aussagekräftigen Beschreibungen hinsichtlich der Kommunikationsschnittstelle eines Dienstes, bspw. in Form von WSDL-Dokumenten, werden ebenfalls weitestgehend vollständig automatisiert als SIB-Komponente bereitgestellt. Ausnahmen müssen hier für komplexe Ein- und Ausgabeinformationen in Form frei definierter Datentypen innerhalb der Dienstbeschreibungen gemacht werden. Diese können durch bestehende Transformationskomponenten aus dem Dienstverzeichnis im Zuge der automatischen Prozesssynthese (siehe Anforderung T1) aufgefangen werden. Je umfangreicher und somit vollständiger die Sammlung solcher SIBs zur Behandlung komplexer Datentypen ausfällt, desto höher die erfolgreiche vollautomatisierte Integration externer Dienste. Web Services ohne formale Beschreibungsinformationen, wie z.B. REST basierte Dienste mit umgangssprachlich formulierten Diensteigenschaften, werden teilautomatisiert und Wizard-gestützt innerhalb von SIBs gekapselt. Diese dritte Form der Integration lässt sich dabei beliebig auf unterschiedliche Klassen von Diensten mit ähnlichen Merkmalen und Repräsentationen erweitern.

S3 - Bereitstellung von Services

Dienste können im Rahmen der Remote Service Umgebung innerhalb der Konfiguration der Service Provider Site über XML Spezifikationen zum Aufruf von Programmcode oder ausführbaren Komponenten auf Seiten des Servers bereitgestellt werden, wobei über eine definierte Schnittstelle unterschiedliche Dienstprotokolle angeboten werden können. Im Kontext der jABC Modellierungsumgebung können zudem vollständige Service Logic Graphen über das Genesys Code Generator Plugin in ausführbaren Programmcode überführt und anschließend automatisiert als jETI Dienst bereitgestellt werden. Über das hierarchische Konzept, welches es ermöglicht SLGs innerhalb von SIBs zu kapseln kann somit zudem ein verschachtelter Aufrufe beliebiger Remote-Dienste unterschiedlicher Granularität erreicht werden.

S4 - Discovery

Die manuelle Suche nach Diensten zur Modellierungszeit bedient sich den Funktionalitäten des jABC auf dem Dienstverzeichnis zur Auffindung entsprechender SIB-Komponenten. Über jETI integrierte Dienste werden automatisch in das lokale Repository übernommen und stehen zur Verfügung. Weitere nicht-lokale Dienste auf Seiten der Service Provider Site können mit Hilfe entsprechender Suchanfragen über das Web-Interface, bzw. einen integrierten Service-Browser gefunden werden. Die automatisierte Suche im Kontext der Prozesssynthese auf Basis von dienstspezifischen Eigenschaften und semantischen Informationen wird mit Hilfe der angebundenen und zu diesem Zweck angepassten miAamics Discovery Engine realisiert.

T1 - Automatisierung

Zur synthetisierten Orchestrierung verteilter Dienste, insbesondere im Kontext von Anforderung S2, bieten die bereitgestellten und integrierten Service-Komponenten eine Schnittstelle zur Annotation von Datenflussinformationen bzgl. der Ein- und Ausgabedaten eines Dienstes. Kombiniert mit als SLG repräsentierten abstrakten Typ- und Aktionstaxonomien auf den entsprechenden Komponenten bilden diese die Informationsgrundlage zur Anbindung und Verwendung adäquater Synthesetechnologien. Im Rahmen dieser Arbeit innerhalb der Fallstudien zur Semantic Web Services Challenge wurden exemplarisch vier Ansätze zur automatischen Komposition von Diensten integriert und insbesondere im Kontext von jETI zur automatischen Integration von SOAP basierten Web Services evaluiert.

T2 - Langlebigkeit

Im Kontext komplexer und rechenintensiver Dienstfunktionalitäten kommt es zu langläufigen Prozessen, welche bei der delegierten Ausführung von jETI Diensten durch die Tracer-Komponente des jABC zu einer Blockade der Ausführung eines Prozesses führen. Um diesem Problem, insbesondere beim Umgang mit großen Datenmengen über ein Netzwerk, entgegenzuwirken, können Dienstausführungen angestoßen und die beteiligten Ein- und Ausgabedaten auf einem verteilten Dateisystem (z.B. WebDAV) hinterlegt werden. Nach erfolgreicher Beendigung des Dienstaufrufes können die Daten von einem Prozess aufgegriffen und weiterverarbeitet werden, ohne dass es zu langwierigen Standzeiten bei der Ausführung eines SLG kommt.

T3 - Semantik

Wie der Analyse von Anforderung T1 bereits zu entnehmen ist, werden Dienste im Kontext der jABC/jETI Umgebung durch die standardisierte Annotation von Datenflussinformationen und Taxonomien ausreichend beschrieben. In der Praxis hat sich dieses Verfahren im Rahmen der vorgestellten Fallstudien bewährt und eine optimale Abstimmung zwischen Flexibilität, Komplexität und einfacher Benutzung gezeigt. Prinzipiell ist es durch das Konzept der SIB-Komponenten innerhalb der Modellierungsumgebung jedoch möglich, einem Dienst beliebige Formen von (semantischen) Informationen zuzuordnen. Gerade im Bezug zu Semantic Web Services können somit von verschiedenen Forschungsgruppen und Communities bevorzugte Konzepte, wie z.B. SAWSDL, umgesetzt und verwendet werden.

T4 - Hands-On Evaluation

Durch die einfache, programmierungsfreie Integration von durch jETI bereitgestellten Diensten (vgl. Anforderung S2) und darüberhinaus unter Berücksichtigung lizenzkritischer Dienstausführungen (vgl. Anforderung S1) hat sich die Kombination des jABC Modellierungs-Frameworks und der jETI Remote Service Umgebung als ausgereifte Test- und Evaluationsplattform beliebiger Dienste und ihrer Funktionalitäten erwiesen. Atomare Bestandteile komplexer und verteilter Gesamtsysteme sowie hochspezialisierte Dienste aus unterschiedlichen Domänen können einfach integriert und von entscheidungskritischen Anwendungsexperten geprüft sowie ausprobiert werden. Die automatische Integration der Dienste unterstützt und beschleunigt diesen Prozess und minimiert die technische Hürde im Umgang mit verteilten Diensten. Über die in Planung befindliche Web-basierte Variante der Modellierungsumgebung und den konzeptuellen Ansatz eines integrierten Testbeds für die von jETI bereitgestellten Dienste, wird dieser Aspekt nochmals verfeinert.

T5 - Ausführbarkeit von Modellen

Die direkte Ausführung von modellierten Prozessen im jABC wird, wie bereits in der Analyse von Anforderung A3 beschrieben, über das Tracer-Plugin realisiert. Ausführungen können dabei hierarchisch verschachtelt auf verschiedenen Ebenen und über multiple Modellierungsprozesse hinaus angestoßen werden. Zur Erzeugung ausführbarer Programme existiert zudem mit dem Genesys-Plugin eine Generierungskomponente innerhalb des jABC, welche die Überführung komplexer Prozessmodelle in nahezu beliebige Zielsprachen ermöglicht und somit eine von der Modellierungsumgebung unabhängige Ausführungsvariante bereitstellt.

Entsprechende Client-Applikationen für Remote-Services und verteilte Dienste selbst können somit den Paradigmen der serviceorientierten Programmierung folgend auf einfache und intuitive Weise erstellt werden.

8.2. Praktische Anwendungen von jETI

Im Kontext der in dieser Arbeit vorgestellten Fallstudien wurden bereits die diversen praktischen Anwendungsfälle und -bereiche herausgestellt, in denen die erarbeitete jETI Remote Service Umgebung nach wie vor eingesetzt wird. Dazu zählen insbesondere

- die Anwendungen im Bereich der Bioinformatik zur Orchestrierung komplexer Analyseprozesse und der Auswertung statistischer Informationen unter dem Oberbegriff **Bio-jETI**,
- die Bereitstellung einer experimentellen Plattform zur Anbindung von Planern und Synthesealgorithmen unter der Überschrift **Plan-jETI**
- sowie die Integration formaler Methoden zum Systementwurf, wie z.B. Dienste aus dem Bereich des Model Checking unter der Bezeichnung **FMICS-jETI**.

Zusammenfassend verdeutlichen diese Arbeitsbereiche und Anwendungsbeispiele die Relevanz des Frameworks, gerade in Bezug auf die Einsatzfähigkeit und Flexibilität im Umgang mit Domänen unterschiedlichster Ausprägungen und Anforderungsdefinitionen hinsichtlich der Integration verteilter Dienste.

9. Ausblick

Unabhängig von den Anfängen der ABC Modellierungsumgebung und des ETI Frameworks ist die jETI Remote Service Umgebung über die letzten sechs Jahre kontinuierlich erweitert und verbessert worden. Insbesondere durch die Neustrukturierung der Architektur und grundlegender Konzepte ist es zu einem offenen und modular aufgebauten Framework gewachsen, welches auch in Zukunft das Potential für weitere Veränderungen mitbringt. In diesem Abschnitt sollen daher Überlegungen hinsichtlich der weiteren Entwicklung von jETI angestrengt und bereits vorhandene Ansätze verbessert und diskutiert werden.

9.1. Interaktion mit grafischen Schnittstellen

Im Zuge erhöhter Bandbreiten der allgemein verfügbaren Netzanbindungen, gewinnt der Ansatz, komplexe Software als verteilten Dienst bereitzustellen (Software as a Service), zunehmend an Relevanz. In der Folge zeigt sich die Tendenz vieler Anbieter, dieses Prinzip bereits heute durch diverse Online-Angebote weiträumig umzusetzen (z.B. Google Docs Office Software[1], etc.). Häufig sind die entsprechenden Werkzeuge zudem mit Schnittstellen zur verteilten Ausführung einzelner Funktionalitäten ausgestattet, so dass einer Integration mit standardisierten und etablierten Mechanismen nichts im Wege steht.
Die Integration von Remote Services mit Hilfe von jETI setzt zum gegenwärtigen Zeitpunkt eine Regelung hinsichtlich der Schnittstellen eines Dienstes voraus. Bereitgestellte Funktionalitäten müssen bspw. als SOAP-basierter Web Service verfügbar sein oder über eine REST Schnittstelle verfügbar gemacht werden. Prinzipiell müssen die Dienste somit in einem ausgewiesenen Format, bzw. einer Technologie vorliegen, die eine direkte Intention für den verteilten Aufruf über ein Netzwerk aufweist. Eine Ausnahme bilden innerhalb von jETI gekapselte Werkzeuge, die als unabhängige Programme auf einer jETI Service Provider Site ausgeführt und mit Hilfe von Wrapper Komponenten als Remote Tool zur Verfügung gestellt werden. Diese einfache Art der verteilten Bereitstellung regulärer und nicht explizit dafür vorbereiteter Software als Service ohne dedizierte Remote Schnittstellen ist eine der herausragenden Eigenschaften der jETI Um-

[1] `http://www.google.com/apps`

gebung. Zur Zeit ist dieser Ansatz jedoch auf Kommandozeilen-basierte Applikationen beschränkt, welche durch den Aufruf mit entsprechender Parametersignatur gesteuert und beeinflusst werden können. Für eine ganzheitliche Umsetzung des Prinzips, allgemein verwendbare Software oder einzelne Ihrer Funktionalitäten als verteilten Dienst bereitzustellen, ist es essentiell, dieses um Methoden zur Integration von Werkzeugen zu erweitern, welche sich auch oder ausschließlich über grafische Benutzeroberflächen steuern lassen. Ähnlich der Konfiguration Kommandozeilen-basierter Applikationen über entsprechende Parametersignaturen muss ein Ansatz ausgearbeitet werden, Befehle in einer grafischen Umgebung ferngesteuert und auf unabhängigen Instanzen der entsprechenden Software abzusetzen. Ein Unterbau mit visuellem Framework auf Seiten der SPS ist daher die Grundvoraussetzung. Dieser kann durch entsprechende Betriebssysteme, bzw. ihrer Erweiterungen hinsichtlich einer GUI gewährleistet werden. Zur Konfiguration einzelner Funktionsschritte in der jeweils zu integrierenden Software können entsprechende Testing-Tools oder visuelle Programmiersprachen eingesetzt werden. Die Ansätze unterscheiden sich dabei grundlegend in der Art ihrer Anwendung:

- Werkzeuge zum **Testen** von Applikationen, wie z.B. IBM Rational Robot[2], arbeiten i.d.R. mit Aufzeichnungen von Befehlsketten, die ein Benutzer auf der grafischen Oberfläche der Software durchführt. Diese Aufzeichnung von Klick-Reihenfolgen funktioniert zuverlässig in abgeschlossenen und hinsichtlich des Bildaufbaus unveränderlichen Umgebungen unter Laborbedingungen. Ändern sich die Randbedingungen, wie. z.B. die Bildschirmauflösung des Server-Systems, kann dies zu Problemen führen. Bezug nehmend auf die parallele Ausführung vieler verschiedener Werkzeuge muss zudem eine entsprechende Anzahl von Instanzen des eingesetzten Test-Werkzeuges in unabhängigen grafischen Umgebungen (Sand-Box) mit jeweils genau einer Applikation ausgeführt werden können.
- **Visuelle Programmiersprachen**, wie z.B. SIKULI[3], bedienen sich der visuellen Analyse des Inhaltes einer grafischen Benutzerumgebung. Beispielsweise können Schaltflächen und Eingabekomponenten auf Basis ihrer optischen Erscheinung identifiziert und angesteuert werden. In der Folge ist eine direkte Interaktion mit dem Interpreter der eingesetzten Sprache möglich. Programmcode, bzw. -skripte können demnach unabhängig von Randbedingungen wie Bildschirmauflösung, etc. ausgeführt werden. Eine vorherige Aufzeichnung von Befehlsketten ist nicht notwendig, da ein direktes Steuern der Applikation zur Ausführungszeit möglich ist. Eine Ausführungsumgebung auf Seiten des Servers kann zudem eine Vielzahl von grafischen Werkzeugen abdecken, da über die Steuerskripte beliebige Applikationen gezielt gestartet und beendet werden können.

[2] `http://www.ibm.com/software/products/de/de/robot/`
[3] `http://sikuli.org/`

9.2. Eigenständige Evaluations- und Testumgebung

Innerhalb der jABC Modellierungsumgebung ist es möglich, über jETI bereitgestellte Dienste einfach und in Kombination mit anderen Diensten zu evaluieren. Über die grafische Benutzeroberfläche können einfache Prozessmodelle erstellt und ausgeführt werden. Die entsprechenden Dienste müssen jedoch zunächst in Form der atomaren Bausteine (SIBs) der Service Logic Graphen vorliegen. Der Evaluierung geht somit die Suche und Selektion einzelner Dienste auf den unterschiedlichen jETI Servern voraus. Eine eigenständige, Web-basierte Evaluations- und Testumgebung auf den jeweiligen Service Provider Sites könnte diesen Prozess deutlich vereinfachen und komfortabler gestalten. Über ein Web-Interface könnten über jETI bereitgestellte Werkzeuge und Web Services direkt angesprochen und ausgeführt werden, was ein unmittelbares Feedback hinsichtlich der Funktionalität noch vor der Integration in das jABC zur Folge hätte. Ebenfalls ist eine Web-basierte Variante der gesamten Modellierungsumgebung denkbar. Letzteres ist mit der Umsetzung einer Online-Ausführungsumgebung für SLG Modelle im Kontext des jABC konkret in Planung, wohingegen erste konzeptionelle Ansätze hinsichtlich eines unabhängigen jETI-Testbed innerhalb dieser Arbeit und der durchgeführten Fallstudien in den vorherigen Abschnitten umrissen sind.

9.3. Cloud Speicher

Zur Persistierung von Eingabe- und Ausgabedaten verwendet jETI lokalen als auch verteilten Speicherplatz auf der Service Provider Site und angeschlossenen WebDAV Dateisystemen. Zur Steigerung der Flexibilität und Kompatibilität sowie zur Erweiterung des Gesamtsystems können aktuelle Cloud-Speicher verwendet werden, um Informationen abzulegen und dauerhaft verfügbar zu machen. Denkbar sind beispielsweise Community-Dienste wie Dropbox[4], box.net[5] oder andere Speicherdienste unterschiedlicher Hersteller. Über komplexe Sicherheits- und Freigabemechanismen ist zudem ein Austausch der Daten über verschiedene Instanzen von jETI Servern oder auch ein unabhängig von der Remote Service Umgebung durchgeführter Austausch von Informationen problemlos möglich. Aktuelle Dienste dieser Art stellen zudem i.d.R. eine umfassende Programmierungsschnittstelle in Form einer API oder REST Diensten bereit, was eine Anbindung vereinfacht. Die funktionale Komponente innerhalb von jETI könnte letzten Endes über die Integration in die Modellierungsumgebung, analog zu der bereits im jABC/jETI modellierten Anbindung der SOAP Web Services, komplett als jETI

[4] `http://www.dropbox.com`
[5] `http://box.net`

Dienst modelliert und umgesetzt werden. Somit kann ergänzend eine eigenständige Nutzung der Speicherdienste auch außerhalb des Kontextes der eigentlichen Remote Service Umgebung innerhalb beliebiger, im jABC modellierter Applikationen erfolgen.

9.4. Media Streaming

Neben dem Aufruf geeigneter Dienste zur Wiedergabe von entfernt persistierten Medien ist eine Integration gängiger Streaming-Technologien, wie z.B. durch die Digital Living Network Alliance (DLNA)[6] spezifiziert, in die jETI Remote Service Umgebung denkbar. Über die Grenzen bestehender Client-Server Strukturen hinweg wäre es möglich, die geräteunabhängige Wiedergabe von Medien unterschiedlicher Beschaffenheit, d.h. Bilder, Audio-Dateien, Videos, Dokumente, etc. in eine bestehende Applikation zu integrieren. Insbesondere in Hinblick auf komplexe multimediale Applikationen ließe sich mit Hilfe geeigneter Technologien die Remote Fähigkeit von jETI weiter ausbauen und steigern.

[6] `http://www.dlna.org/`

Literaturverzeichnis

[A+01] Ankolenkar, A. u. a.: *DAML-S: Semantic Markup For Web Services.* 2001. – `http://www.daml.org/services/daml-s/2001/10/daml-s.html`

[Ach10] Achilleos, A.: *Model-driven Petri Net based Framework for Pervasive Service Creation.*, School of Computer Science and Electronic Engineering, University of Essex, UK, Diss., 2010

[AKP11] Achilleos, A. ; Kapitsaki, G. M. ; Papadopoulos, G. A.: A Model-Driven Framework for Developing Web Service Oriented Applications. In: *Proceedings of the 11th International Conference on Web Engineering (ICWE) - 7th Model-Driven Web Engineering Workshop (MDWE)* Bd. 6757. Paphos, Zypern : Springer, Juni 2011 (Lecture Notes in Computer Science (LNCS)). – ISBN 978–3–642–22232–0, S. 20–24

[Bar77] Barwise, J. ; Barwise, J. (Hrsg.): *An introduction to first-order logic.* 1977. – ISBN 978–0–444–86388–1

[BCFJ08] Binder, W. ; Constantinescu, I. ; Faltings, B. ; Jurca, R.: Automating the Creation of Compound Web Applications. In: *ERCIM News* 72 (2008), Januar. – `http://ercim-news.ercim.eu/automating-the-creation-of-compound-web-applications`

[BCM+03] Baader, F. ; Calvanese, D. ; McGuinness, D. L. ; Nardi, D. ; Patel-Schneider, P. F.: *The Description Logic Handbook: Theory, Implementation, Applications.* Cambridge, UK : Cambridge University Press, 2003. – ISBN 0–521–78176–0

[BCV+08] Brambilla, M. ; Ceri, S. ; Valle, E. D. ; Facca, F.M. ; Kubczak, C. ; Margaria, T. ; Steffen, B. ; Winkler, C.: Comparison: Mediation on WebML/WebRatio and jABC/jETI. In: Petrie, C. (Hrsg.) ; Margaria, T. (Hrsg.) ; Zaremba, M. (Hrsg.) ; Lausen, H. (Hrsg.): *Semantic Web Services Challenge – Results from the First Year.* Springer Verlag, 2008. – ISBN 978–0–387–72495–9, S. 153–166

[BDR08] Bergel, A. ; Ducasse, S. ; Renggli, L.: Seaside - Advanced Composition and Control Flow for Dynamic Web Applications. In: *ER-*

CIM News 72 (2008), Januar. – http://ercim-news.ercim.eu/content/view/325/536/

[BEK+00] Box, D. ; Ehnebuske, D. ; Kakivaya, G. ; Layman, A. ; Mendelsohn, N. ; Nielsen, H. F. ; Thatte, S. ; Winer, D.: *Simple Object Access Protocol (SOAP).* Mai 8 2000. – http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[BK08] Baier, C. ; Katoen, J. P.: *Principles of model checking.* MIT Press, 2008. – ISBN 978–0–262–02649–9

[BLFF96] Berners-Lee, T. ; Fielding, R. ; Frystyk, H.: *Hypertext Transfer Protocol (RFC 1945).* Mai 1996. – http://tools.ietf.org/html/rfc1945

[BMRS07] Bakera, M. ; Margaria, T. ; Renner, C.D. ; Steffen, B.: Property-driven functional healing: Playing against undesired behavior. In: Schieferdecker, I. (Hrsg.) ; Goericke, S. (Hrsg.): *Business Process Engineering, Proceedings of the CONQUEST 2007*, 2007, S. 363–372

[BMW97] Braun, V. ; Margaria, T. ; Weise, C.: Integrating Tools in the ETI Platform. In: *International Journal on Software Tools for Technology Transfer (STTT)* 1 (1997), Nr. 2, S. 31–48

[bpe02] IBM: *Business Process Execution Language for Web Services version 1.1.* Juli 2002. – http://www.ibm.com/developerworks/library/specification/ws-bpel/

[CCMW01] Christensen, E. ; Curbera, F. ; Meredith, G. ; Weerawarana, S.: *Web Services Description Language (WSDL).* März 15 2001. – http://www.w3.org/TR/wsdl.html

[Cer95] Cerf, V.: *An Agreement between the Internet Society and Sun Microsystems, Inc. in the Matter of ONC RPC and XDR Protocols (RFC 1790).* April 1995. – http://www.networksorcery.com/enp/rfc/rfc1790.txt

[CFB+02] Ceri, S. ; Fraternali, P. ; Bongio, A. ; Brambilla, M. ; Comai, S. ; Matera, M.: *Designing Data-Intensive Web Applications.* Morgan Kauffmann, 2002. – ISBN 1558608435

[CGP00] Clarke, E. M. ; Grumberg, O. ; Peled, D. A.: *Model Checking.* 2000. – ISBN 0–262–03270–8

[Chi99] Chiu, A.: *Authentication Mechanisms for ONC RPC (RFC 2695).* September 1999. – http://www.networksorcery.com/enp/rfc/rfc2695.txt

[Chu63] Church, A.: Logic, arithmetic and automata. In: *In Proceedings of the International Congress of Math*, Almqvist and Wiksells, 1963, S. 23–35

[CM08] Crane, D. ; McCarthy, P.: *Comet and Reverse Ajax: The Next-Generation Ajax 2.0*. Apress, 2008. – ISBN 978–1590599983

[Con71] *Kapitel* 4. In: Conway, J. H.: *Regular algebra and finite machines*. Chapman and Hall, 1971. – ISBN 0–412–10620–5

[con02a] consortium, OASIS: *OASIS UDDI Specifications*. Juli 19 2002. – `http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm`

[con02b] consortium, OASIS: *OASIS/ebXML Registry Services Specification v2.0*. April 2002. – `http://www.oasis-open.org/committees/regrep/documents/2.0/specs/ebrs.pdf`

[con03] consortium, OASIS: *OASIS Web Services Business Process Execution Language (WSBPEL) TC*. 2003. – `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel`

[con11] consortium, OMG: *OMG Unified Modeling Language (OMG UML), Infrastructure*. 2011. – `http://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF/`

[doc08] Technische Universität Dortmund: *Projektarbeit: DoCamping - ConnectIT*. 2008. – `http://connectit.cs.uni-dortmund.de/`

[Doe06] Doedt, M.: *Erweiterung der jABC Framework Bibliothek um eine durch entsprechenden Hotspot individuell anpassbare Ausführungsschicht*, Technische Universität Dortmund, Diplomarbeit, Mai 2006

[Eis06] Eisler, M.: *XDR: External Data Representation Standard (RFC 4506)*. Mai 2006. – `http://tools.ietf.org/html/rfc4506`

[FB02] Fensel, D. ; Bussler, C.: The web service modeling framework WSMF. In: *Electronic Commerce Research and Applications* 1 (2002), Nr. 2, S. 113–137

[FH+97] Finnigan, P. J. ; Holt, R. C. u. a.: The software bookshelf. In: *IBM Systems Journal* 36 (1997), Nr. 4, S. 564–593. – ISSN 0018–8670

[Fie00] Fielding, Roy T.: *Architectural Styles and the Design of Network-based Software Architectures*, University of California, Diss., 2000. – `http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm`

[FL06] Farrell, J. ; Lausen, H.: *Semantic Annotations for WSDL*

(SAWSDL). April 2006. – http://www.w3.org/TR/sawsdl/

[FLP+06] Fensel, D. ; Lausen, H. ; Polleres, A. ; Bruijn, J. de ; Stollberg, M. ; Roman, D. ; Domingue, J.: *Enabling Semantic Web Services: The Web Service Modeling Ontology*. Secaucus, NJ, USA : Springer-Verlag, New York, Inc., 2006

[Fun09] Funke, H.: Das OSGi Framework. In: *IT Republic* JAXenter (2009). – http://it-republik.de/jaxenter/artikel/Das-OSGi-Framework-2221.html

[Gar05] Garrett, J. J.: Ajax: A New Approach to Web Applications. In: *Adaptive Path* Ideas (2005), Februar 18. – http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications

[Gra10] Gravelle, R.: Comet Programming: Using Ajax to Simulate Server Push. In: *Webreference.com* (2010). – http://www.webreference.com/programming/javascript/rg28/index.html

[Gro98] Group, The O.: *DCE Today: An Indispensable Guide to DCE*. Prentice Hall, 1998. – ISBN 0135756979

[gro11a] group, OMG: *Business Process Model and Notation (BPMN)*. 2011. – http://www.omg.org/spec/BPMN/2.0/PDF/

[gro11b] group, OMG: *Common Object Request Broker Architecture (CORBA) Specification, Version 3.2 Part 1: CORBA Interfaces* . November 2011. – http://www.omg.org/spec/CORBA/3.2/Interfaces/PDF/

[gro11c] group, OMG: *Common Object Request Broker Architecture (CORBA) Specification, Version 3.2 Part 2: CORBA Interoperability*. November 2011. – http://www.omg.org/spec/CORBA/3.2/Interoperability/PDF/

[gro11d] group, OMG: *Common Object Request Broker Architecture (CORBA) Specification, Version 3.2 Part 3: CORBA Component Model*. November 2011. – http://www.omg.org/spec/CORBA/3.2/Components/PDF/

[HCM+05] Haller, A. ; Cimpian, E. ; Mocan, A. ; Oren, E. ; Bussler, C.: WSMX - A Semantic Service-Oriented Architecture. In: *Proceedings of the IEEE International Conference on Web Services (ICWS)*. Washington, DC, USA : IEEE Computer Society, 2005, S. 321–328

[Hor51] Horn, A.: On sentences which are true of direct unions of algebras. In: *Journal of Symbolic Logic* 16 (1951), S. 14–21

[HTC99] Hunt, A. ; Thomas, D. ; Cunningham, W.: *The Pragmatic Programmer: From Journeyman to Master*. Addison-Wesley Longman, 1999. – 26 S. – ISBN 978–0–201–61622–4

[Joa11] Joachim, Nils: A Literature Review of Research on Service-Oriented Architectures (SOA): Characteristics, Adoption Determinants, Governance Mechanisms, and Business Impact. In: *AMCIS*, 2011, S. 3001–3011. – `http://aisel.aisnet.org/amcis2011_submissions/339`

[Joe11] Joerges, S.: *Genesys: A Model-Driven and Service-Oriented Approach to the Construction and Evolution of Code Generators*, Technische Universität Dortmund, Diss., 2011

[Kai07] Kaiser, M.: Towards the realization of policy-oriented enterprise management. In: *IEEE Computer Society Special Issue on Service-Oriented Architecture forthcoming* 11 (2007), S. 65–71

[KJMS09] Kubczak, C. ; Jörges, S. ; Margaria, T. ; Steffen, B.: eXtreme Model-Driven Design with jABC. In: Vogel, R. (Hrsg.): *Proceedings of the Tools and Consultancy Track of the Fifth European Conference on Model-Driven Architecture Foundations and Applications (ECMDA-FA)* Bd. WP09-12. P.O. Box 217, 7500 AE Enschede, The Netherlands : University of Twente, Centre for Telematics and Information Technology (CTIT), Juni 25-26 2009 (CTIT proceedings). – ISSN 0929–0672, S. 78–99. – `http://www.ctit.utwente.nl/`

[KL07] Kaiser, M. ; Lemcke, J.: Towards a framework for policy-oriented enterprise management. In: *AAAI* (2007)

[KMFS06] Kubczak, C. ; Margaria, T. ; Fritsch, A. ; Steffen, B.: Biological LC/MS Preprocessing and Analysis with jABC, jETI and xcms. In: *Proceedings of the 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA)*. Paphos, Zypern : IEEE Computer Society Press, November 15-19 2006, S. 308–313

[KMK+08] Kubczak, C. ; Margaria, T. ; Kaiser, M. ; Lemcke, J. ; Knuth, B. ; García-Castro, R. (Hrsg.) ; Gómez-Pérez, A. (Hrsg.) ; Petrie, C. J. (Hrsg.) ; Valle, E. D. (Hrsg.) ; Küster, U. (Hrsg.) ; Zaremba, M. (Hrsg.) ; Shafiq, M. O. (Hrsg.): *Abductive Synthesis of the Mediator Scenario with jABC and GEM*. Teneriffa, Spain, Juni 1-2 2008 (CEUR Workshop Proceedings). – `http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-359/`

[KMS+08] Kubczak, C. ; Margaria, T. ; Steffen, B. ; Winkler, C. ; Hun-

GAR, H.: An Approach to Discovery with miAamics and jABC. In: PETRIE, C. (Hrsg.) ; MARGARIA, T. (Hrsg.) ; ZAREMBA, M. (Hrsg.) ; LAUSEN, H. (Hrsg.): *Semantic Web Services Challenge – Results from the First Year*. Springer Verlag, 2008. – ISBN 978–0–387–72495–9, S. 217–234

[KMSN08] KUBCZAK, C. ; MARGARIA, T. ; STEFFEN, B. ; NAGEL, R.: Service-oriented Mediation with jABC/jETI. In: PETRIE, C. (Hrsg.) ; MARGARIA, T. (Hrsg.) ; ZAREMBA, M. (Hrsg.) ; LAUSEN, H. (Hrsg.): *Semantic Web Services Challenge – Results from the First Year*. Springer Verlag, 2008. – ISBN 978–0–387–72495–9, S. 71–99

[KNS92] KELLER, G. ; NÜTTGENS, M. ; SCHEER, A. W.: Semantische Prozeßmodellierung auf der Grundlage Ereignisgesteuerter Prozeßketten (EPK). In: *Veröffentlichungen des Instituts für Wirtschaftsinformatik* 89 (1992)

[Koz83] KOZEN, Dexter: Results on the Propositional mu-Calculus. In: *Theoretical Computer Science* 27 (1983), S. 333–354

[Kri63] KRIPKE, S.: Semantical Considerations on Modal Logic. In: *Acta Philosophica Fennica* 16 (1963), S. 83–94

[KS86] KOWALSKI, R. ; SERGOT, M.: A Logic-Based Calculus of Events. In: *New Generation Computing* 4 (1986), S. 67–95

[Kub05] KUBCZAK, C.: *Entwicklung einer verteilten Umgebung zur Personalisierung von Web-Applikationen*, Technische Universität Dortmund, Diplomarbeit, 2005

[Kub08] KUBCZAK, C.: *Agile Multimodal Emergency Handling with Telecommunication- and GIS-Mashups*. November 6-7 2008. – `http://www.fokus.fraunhofer.de/en/fokus_events/ngni/ims_ws_08/index.html`

[KVW+08] KUBCZAK, C. ; VITVAR, T. ; WINKLER, C. ; ZAHARIA, R. ; ZAREMBA, M.: Comparison: Discovery on WSMOLX and miAamics/jABC. In: PETRIE, C. (Hrsg.) ; MARGARIA, T. (Hrsg.) ; ZAREMBA, M. (Hrsg.) ; LAUSEN, H. (Hrsg.): *Semantic Web Services Challenge – Results from the First Year*. Springer Verlag, 2008. – ISBN 978–0–387–72495–9, S. 247–262

[Ley01] LEYMANN, F.: *Web Service Flow Language (WSFL 1.0)*. Mai 2001. – `http://www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf`

[Llo87] LLOYD, J. W.: *Foundations of Logic Programming*. 2. Springer-

Verlag, 1987

[LNMS09] Lamprecht, A.-L. ; Naujokat, S. ; Margaria, T. ; Steffen, B.: *Semantics-Based Composition of EMBOSS Services with Bio-jETI*. November 20 2009. – `http://ceur-ws.org/Vol-559/`

[M+04] Martin, D. u. a.: *OWL-S: Semantic Markup for Web Services*. November 22 2004. – `http://www.w3.org/Submission/OWL-S/`

[Mar05] Margaria, T.: Web Services-Based Tool-Integration in the ETI Platform. In: *SoSyM, Int. Journal on Software and System Modelling* 4 (2005), Mai, Nr. 2, S. 141–156

[MBK97] Margaria, T. ; Braun, V. ; Kreileder, J.: Interacting with ETI: A User Session. In: *Int. Journal on Software Tools for Technology Transfer (STTT)* 1 (1997), Nr. 2, S. 49–63

[MBK+08] Margaria, T. ; Bakera, M. ; Kubczak, C. ; Naujokat, S. ; Steffen, B.: Automatic Generation of the SWS-Challenge Mediator with jABC/ABC. In: Petrie, C. (Hrsg.) ; Margaria, T. (Hrsg.) ; Zaremba, M. (Hrsg.) ; Lausen, H. (Hrsg.): *Semantic Web Services Challenge – Results from the First Year*. Springer Verlag, 2008. – ISBN 978–0–387–72495–9, S. 119–138

[MH69] McCarthy, J. ; Hayes, P.: Some philosophical problems from the standpoint of artificial intelligence. In: *Machine Intelligence* 4 (1969), S. 463–502

[Mic12] Microsoft: *XAML Overview (WPF)*. 2012. – `http://msdn.microsoft.com/en-us/library/ms752059(v=vs.110).aspx`

[Mis04] Misra, J.: A Programming Model for the Orchestration of Web Services. In: *Proceedings of the Second International Conference on Software Engineering and Formal Methods (SEFM)*, IEEE, September 28-30 2004. – ISBN 0–7695–2222–X, S. 2–11

[MKS08] Margaria, T. ; Kubczak, C. ; Steffen, B.: Bio-jETI: A Service Integration, Design, and Provisioning Platform for Orchestrated Bioinformatics Processes. In: *BioMed Central (BMC) Bioinformatics Supplement dedicated to Network Tools and Applications in Biology Workshop (NETTAB)* 9 (2008), Nr. S-4, S. 12. – `http://www.biomedcentral.com/1471-2105/9?issue=S4`

[MKSN06] Margaria, T. ; Kubczak, C. ; Steffen, B. ; Naujokat, S.: The FMICS-jETI Platform: Status and Perspectives. In: *Proceedings of the 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA)*. Paphos, Zypern

: IEEE Computer Society Press, November 15-19 2006, S. 414–418

[MMCZ06] Mocan, A. ; Moran, M. ; Cimpian, E. ; Zaremba, M.: Filling the Gap. Extending Service Oriented Architectures with Semantics. In: *Proceedings of the ICEBE* IEEE Computer Society, 2006, S. 594–601

[MMK+09] Margaria, T. ; Meyer, D. ; Kubczak, C. ; Isberner, M. ; Steffen, B.: Synthesizing Semantic Web Service Compositions with jMosel and Golog. In: *Proceedings of the 8th International Semantic Web Conference (ISWC)* Bd. 5823. Washington D.C., United States : Springer Verlag, Oktober 25-29 2009 (Lecture Notes in Computer Science, LNCS), S. 392–407

[MNS05] Margaria, T. ; Nagel, R. ; Steffen, B.: jETI: A Tool for Remote Tool Integration. In: *TACAS*, 2005, S. 557–562

[MOSS99] Müller-Olm, M. ; Schmidt, D. A. ; Steffen, B.: Model-Checking: A Tutorial Introduction. In: *Proceedings of the 6th International Symposium on Static Analysis (SAS)*, Springer, 1999, S. 330–354

[MS04] Margaria, T. ; Steffen, B.: Lightweight coarse-grained coordination: a scalable system-level approach. In: *STTT* 5 (2004), Nr. 2-3, S. 107–123. – `http://www.springerlink.com/index/10.1007/s10009-003-0119-4`

[MS06] Margaria, T. ; Steffen, B.: Service Engineering: Linking Business and IT. In: *IEEE Computer, issue 60th anniv. of the Computer Society* (2006), Oktober, S. 53–63

[MS08] Margaria, T. ; Steffen, B.: Agile IT: Thinking in User-Centric Models. In: *Proceedings of the International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA)*, 2008, S. 490–502. – `http://www.springerlink.com/content/w6gl6mqu0711g464/`

[MS09] Margaria, T. ; Steffen, B.: *Handbook of Research on Business Process Modeling*. IGI Global, 2009. – 1–26 S.

[MSC+01] Mancoridis, S. ; Souder, T. S. ; Chen, Y. ; Gansner, E. R. ; Korn, J. L.: REportal: A web-based portal site for reverse engineering. In: *In Proceedings of the 8th Working Conference on Reverse Engineering (WCRE)*. Washington D.C., USA : IEEE, 2001. – ISBN 0–7695–1303–4, S. 221

[Mus09] Musil, C. von: *JR - Integration von Statistikfunktionalität in eine Prozessmanagementumgebung*, Technische Universität Dortmund,

Diplomarbeit, 2009

[Nag09] Nagel, R.: *Technische Herausforderungen modellgetriebener Beherrschung von Prozesslebenszyklen aus der Fachperspektive: Von der Anforderungsanalyse zur Realisierung*, Technische Universität Dortmund, Diss., 2009

[Nau09] Naujokat, S.: *Automatische Generierung von Prozessen im jABC*, Technische Universität Dortmund, Diplomarbeit, 2009

[OAF+04] Oinn, T. ; Addis, M. ; Ferris, J. u. a.: Taverna: a tool for the composition and enactment of bioinformatics workflows. In: *Bioinformatics* 20 (2004), Nr. 17, S. 3045–3054

[Ora12] Oracle: *Remote Method Invocation*. 2012. – `http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136424.html`

[owl04] World Wide Web Consortium (W3C): *OWL Web Ontology Language Semantics and Abstract Syntax*. November 10 2004. – `http://www.w3.org/TR/owl-semantics/`

[Pap06] Papazoglou, M. P.: *Principles and Foundations of Web Services: An Holistic View*. Addison-Wesley, 2006

[Pet62] Petri, C. A.: *Kommunikation mit Automaten*, Institut für instrumentelle Mathematik der Universität Bonn, Diss., 1962

[Pos82] Postel, J. B.: *Simple Mail Transfer Protocol (SMTP) (RFC 821)*. August 1982. – `http://tools.ietf.org/html/rfc821`

[PR85] Postel, J. B. ; Reynolds, J.: *File Transfer Protocol (FTP) (RFC 959)*. Oktober 1985. – `http://tools.ietf.org/html/rfc959`

[Rei91] Reiter, R.: The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. (1991), S. 359–380

[RKL+05] Roman, D. ; Keller, U. ; Lausen, H. ; Bruijn, J. de ; Lara, R. ; Stollberg, M. ; Polleres, A. ; Feier, C. ; Bussler, C. ; Fensel, D.: Web Service Modeling Ontology. In: *Applied Ontologies* 1 (2005), Nr. 1, S. 77–106

[RLB00] Rice, P. ; Longden, I. ; Bleasby, A.: EMBOSS: The European Molecular Biology Open Software Suite. In: *Trends in Genetics* 16 (2000), Juni, Nr. 6, S. 276–277

[rpc76] *A High-Level Framework for Network-Based Resource Sharing (RFC 707)*. Januar 14 1976. – `http://tools.ietf.org/html/rfc707`

[rpc88] SUN Microsystems: *RPC: Remote Procedure Call Protocol Specification Version 2 (RFC 1057)*. Juni 1988. – `http://www.ietf.org/rfc/rfc1057.txt`

[rpc04] ETH Zürich: *Vorlesungsskript: Verteilte Systeme*. 2004. – `http://www.vs.inf.ethz.ch/edu/WS0405/VS/Vorl.VertSys04_05-4.pdf`

[SAP12a] SAP: *ABAP Programmierung (BC-ABA)*. 2012. – `http://help.sap.com/saphelp_470/helpdata/de/d3/2e974d35c511d1829f0000e829fbfe/frameset.htm`

[SAP12b] SAP: *SAP Java Connector*. 2012. – `http://help.sap.com/saphelp_nwpi711/helpdata/de/48/70792c872c1b5ae10000000a42189c/content.htm`

[SMB97] Steffen, B. ; Margaria, T. ; Braun, V.: The Electronic Tool Integration platform: concepts and design. In: *International Journal on Software Tools for Technology Transfer (STTT)* 1 (1997), November, S. 9–30

[SMN05] Steffen, B. ; Margaria, T. ; Nagel, R.: Remote Integration and Coordination of Verification Tools in jETI. In: *Proceedings of the 12th IEEE International Conference on the Engineering of Computer Based Systems (ECBS)*. Greenbelt, USA : IEEE Computer Society Press, April 2005, S. 431–436

[SMN+06] Steffen, B. ; Margaria, T. ; Nagel, R. ; Jörges, S. ; Kubczak, C.: Model-Driven Development with the jABC. In: *Proceedings of the 2nd international Haifa verification conference on Hardware and software, verification and testing (HVC)* Bd. 4383 IBM, Springer Verlag, Oktober 2006 (Lecture Notes in Computer Science, LNCS), S. 92–108

[Sri95a] Srinivasan, R.: *Binding Protocols for ONC RPC Version 2 (RFC 1833)*. August 1995. – `http://www.networksorcery.com/enp/rfc/rfc1833.txt`

[Sri95b] Srinivasan, R.: *RPC: Remote Procedure Call Protocol Specification Version 2 (RFC 1831)*. 1995. – `http://web.mit.edu/rfc/rfc1831.txt`

[tav12] *Taverna REST API Dokumentation*. 2012. – `http://www.taverna.org.uk/documentation/taverna-2-x/server/2-2/rest-api/`

[TWMS06] Topnik, C. ; Wilhelm, E. ; Margaria, T. ; Steffen, B.: jMosel: A Stand-Alone Tool and jABC Plugin for M2L(Str). In: *Proceedings of Model Checking Software 13th International SPIN Workshop* Bd.

3925, 2006 (Lecture Notes in Computer Science (LNCS)), S. 293–298

[VCC05] Valle, E. D. ; Cerizza, D. ; Celino, I.: The mediators centric approach to auto- matic web service discovery of glue. In: Hepp, M. (Hrsg.) ; Polleres, A. (Hrsg.) ; Harmelen, F. van (Hrsg.) ; Genesereth, M. R. (Hrsg.): *MEDIATE CEUR Workshop Proceedings* Bd. 168. Amsterdam, The Netherlands : CEUR-WS.org, Dezember 2005, S. 35–50. – `http://CEUR-WS.org/Vol-168/`

[VMK+07] Vitvar, T. ; Mocan, A. ; Kerrigan, M. ; Zaremba, M. ; Moran, M. ; Cimpian, E. ; Haselwanter, T. ; Fensel, D.: Semantically-enabled service oriented architecture: Concepts, technology and application. In: *Service Oriented Computing and Applications* 1 (2007), Nr. 2

[(W311] (W3C), World Wide Web C.: *Web Services Activity*. April 28 2011. – `http://www.w3.org/2002/ws/`

[(W312] (W3C), World Wide Web C.: *Extensible Markup Language (XML)*. Januar 24 2012. – `http://www.w3.org/XML/`

[web07] Dusseault, L. (Hrsg.): *HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)*. Juni 2007. – `http://tools.ietf.org/html/rfc4918`

[WH66] Wirth, N. ; Hoare, C. A. R.: A contribution to the development of ALGOL. In: *Communications of the ACM* 9 (1966), S. 413–432

[WHK97] Wahl, M. ; Howes, T. ; Kille, S.: *Lightweight Directory Access Protocol (v3)*. Dezember 1997. – `http://tools.ietf.org/html/rfc2251`

[Win99] Winer, D.: *XML-RPC Specification*. Juni 15 1999. – `http://xmlrpc.scripting.com/spec.html`

[xdr87] SUN Microsystems: *XDR: External Data Representation Standard (RFC 1014)*. Juni 1987. – `http://tools.ietf.org/html/rfc1014`

[XZ07] Xiao, J. ; Zheng, L.: A Multi-Stage Modeling Framework for Web Service Composition. In: *International Conference on Industrial Engineering and Engineering Management (IEEM)*. Singapur : IEEE, Dezember 2-4 2007. – ISBN 978–1–4244–1529–8, S. 1782–1786

[YK01] Yang, G. ; Kifer, M.: *FLORA-2: User's Manual*. 2001. – `http://flora.sourceforge.net/visualizer/manual.pdf`

Abkürzungsverzeichnis

ABC	Agent Building Center
ADD	Algebraic Decision Diagram
AJAX	Asynchronous Javascript and XML
API	Application Programming Interface
ASM	Abstract State Machine
AXIS	Apache eXtensible Interaction System
BPEL	Business Process Execution Language
BPEL4WS	Business Process Execution Language for Web Services
BPMN	Business Process Modeling Notation
CASE	Computer Aided Software Engineering
COC	Convention over Configuration
CORBA	Common Object Request Broker Architecture
CS	Component Server
CTL	Computational Tree Logic
CVS	Concurrent Versions System
DAML-S	DARPA agent markup language for services
DCE	Distributed Computing Environment
DCOM	Distributed Component Object Model
DL	Description Logic
DLNA	Digital Living Network Alliance
DRY	Don't Repeat Yourself
DSL	Domain Specific Language
DSM	Domain Specific Modeling
ebXML	Electronic Business using XML
EMF	Eclipse Modeling Framework
EPK	Ereignisgesteuerte Prozesskette
ERCIM	European Research Consortium for Informatics and Mathematics

ERP	Enterprise Resource Planning
ETI	Electronic Tool Integration
FMICS	Formal Methods for Industrial Critical Systems
FOL	First Order Logic
FTP	File Transfer Protocol
GEM	Goal-oriented Enterprise Management
GUI	Graphical User Interface
GWT	Google Web Toolkit
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IRIS	Integrated Rule Inference System
jABC	Java Application Building Center
JAX-RS	Java API for RESTful Web Services
JAX-WS	Java API for XML Web Services
jBPM	JBoss Business Process Management
JCo	Java Connector
jETI	Java Electronic Tool Integration
KS	Kripke Struktur
KTS	Kripke Transitionssystem
LC/MS	Liquid Chromatography / Mass Spectrometry
LDAP	Lightweight Directory Access Protocol
LP	Logic Programming
LRT	Long Running Transaction
LTS	Labeled Transition System
M2L(Str)	Monadic Second-Order Logic on Strings
MDSD	Model-Driven Software Development
ONC	Open Network Computing
OSGI	Open Services Gateway Initiative
OWL	Web Ontology Language
OWL-S	Web Ontology Language for Web Services
PML	Presentation Modeling Language
POC	Proof of Concept
POI	Points of Interest
PROPHETS	Process Realization and Optimization Platform using a Human-readable Expression of Temporal-logic Synthesis

QOS ..... Quality of Service
RDF ..... Resource Description Framework
REST ..... Representational State Transfer
RMI ..... Remote Method Invocation
RPC ..... Remote Procedure Call
RPyC ..... Remote Python Call
RSS ..... Really Simple Syndication
SAAS ..... Software as a Service
SAWSDL ..... Semantic Annotations for WSDL
SCUFL ..... Simple Conceptual Unified Flow Language
SIB ..... Service Independent Building Block
SLA ..... Service Level Agreement
SLG ..... Service Logic Graph
SLTL ..... Semantic Linear Time Logic
SMS ..... Short Message Service
SMTP ..... Simple Mail Transfer Protocol
SOA ..... Service Oriented Architecture
SPS ..... Service Provider Site
SSL ..... Secure Sockets Layer
SSO ..... Single Sign On
SVN ..... Apache Subversion
SWSC ..... Semantic Web Services Challenge
TCP ..... Transmission Control Protocol
TLS ..... Transport Layer Security
UDDI ..... Universal Description, Discovery and Integration
UDP ..... User Datagram Protocol
UI ..... User Interface
UML ..... Unified Modeling Language
URI ..... Uniform Resource Identifier
VFS ..... Virtual File System
WebDAV ..... Web-based Distributed Authoring and Versioning
WebML ..... Web Modeling Language
WSDL ..... Web Service Description Language
WSFL ..... Web Service Flow Language
WSMF ..... Web Service Modeling Framework
WSMO ..... Web Service Modeling Ontology

WSMT ..... Web Service Modeling Toolkit
WSMX ..... Web Service Execution Language
XAML ..... Extensible Application Markup Language
XDR ..... External Data Representation
XHTML ..... Extensible Hypertext Markup Language
XMDD ..... Extreme Model Driven Development
XML ..... Extensible Markup Language

Abbildungsverzeichnis

Tabellenverzeichnis

Sachverzeichnis

Z