Ontologiebasierte Applikationsintegration auf Nutzerschnittstellenebene

Heiko Paulheim
FG Knowledge Engineering
Technische Universität Darmstadt
Hochschulstraße 10
64289 Darmstadt
paulheim@ke.tu-darmstadt.de

Abstract: Typische IT-Landschaften bestehen aus verschiedenen Anwendungen, die parallel genutzt werden. Die Kombination solcher Anwendungen zu integrierten Systemen ist nicht trivial, insbesondere dann nicht, wenn die bestehenden Nutzerschnittstellen weiter verwendet werden sollen. Existierende Ansätze wie Portal- oder Mashup-Lösungen sind vielfach nicht flexibel genug, um eine nahtlose Integration unter Wahrung des Paradigmas der losen Kopplung zu implementieren. Diese Dissertation stellt einen Ansatz vor, der Ontologien und Reasoning nutzt, um Applikationen auf Nutzerschnittstellenebene zu integrieren. Die Arbeit zeigt auf, wie sowohl technologische als auch konzeptionelle Heterogenitäten zwischen den integrierten Applikationen durch den Einsatz von Ontologien überwunden werden können, und diskutiert eine effiziente Implementierung der semantik- und regelgestützten Verarbeitung von Nachrichten zwischen Applikationen. Darüber hinaus wird aufgezeigt, wie der Nutzer bei der Interaktion mit derart integrierten System zusätzlich unterstützt werden kann. Die gesamte Arbeit wird in einer begleitenden Fallstudie aus dem Bereich des Katastrophenmanagements auf ihre praktische Umsetzbarkeit hin geprüft.

1 Einleitung

Applikationsintegration bezeichnet das Zusammenfügen bestehender Anwendungen oder Teile davon zu einem neuen Gesamtsystem, das über die Möglichkeiten der Einzelsysteme hinaus neue Funktionalitäten bereitstellt. Folgt man der klassischen Dreiteilung von Softwaresystemen in Datenhaltung, Applikationslogik und Nutzerschnittstelle [Fow03], so existieren prinzipiell drei Möglichkeiten, eine solche Integration umzusetzen: Integration auf der Ebene der Datenhaltung, auf der Ebene der Applikationslogik, und auf der Ebene der Nutzerschnittstelle [DYB+07].

Wird ein Ansatz auf der Ebene der Datenhaltung oder der Applikationslogik gewählt, so wird stets zumindest eine neue Nutzerschnittstelle implementiert (siehe Abb. 1). Daher haben Ansätze auf der Ebene der Nutzerschnittstelle sowohl für den Entwickler als auch für den Endbenutzer des integrierten Systems. Auf die Nutzerschnittstelle eines interaktiven Systems entfallen ca. 50% des gesamten Entwicklungsaufwandes [MR92], der Wiederverwendungsgrad ist also vielfach höher, wenn ein Integrationsansatz auf der Ebene der

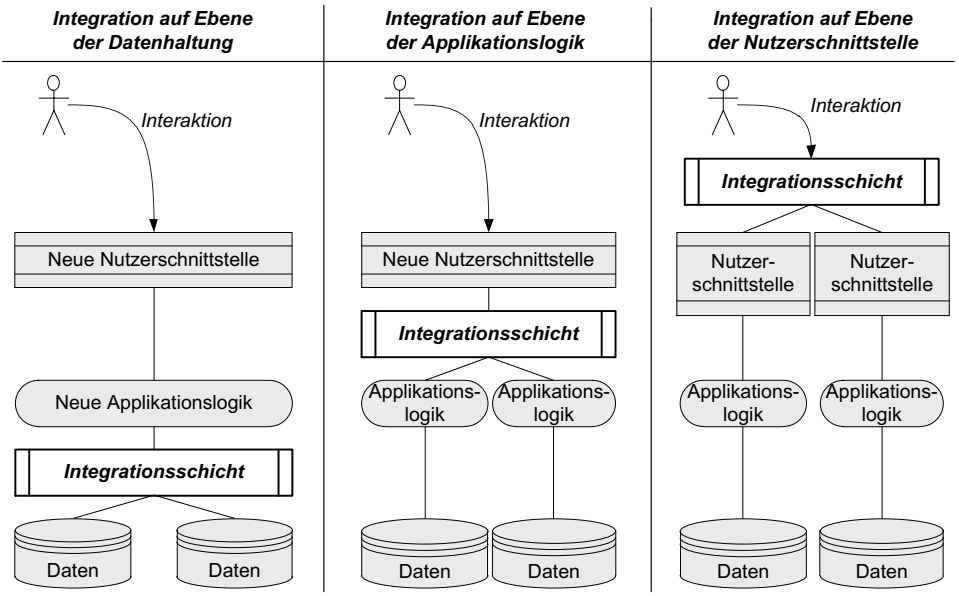

Abbildung 1: Ansätze für Applikationsintegration auf unterschiedlichen Schichten [PP10], angelehnt an [DYB+07]

Nutzerschnittstelle gewählt wird. Außerdem können Nutzer bei derart integrierten Systemen mit bereits bekannten Nutzerschnittstellen interagieren, was die Einarbeitungszeit erheblich verkürzt.

Derzeit verfügbare Ansätze zur Applikationsintegration auf Nutzerschnittstellenebene sind *Portal*- und *Mashup*- Lösungen. Beide erlauben die Wiederverwendung kompletter Applikationen inklusive ihrer Nutzerschnittstellen und die Kombination zu neuen Systemen. Während Portal-Lösungen eher an professionelle Entwickler gerichtet sind und zusätzliche Funktionalitäten wie z.B. Single-Sign-On bieten, sind Mashup-Lösungen eher auf technisch versierte Endbenutzer und semiprofessionelle Programmierer ausgelegt. Untersucht man bestehende Produkte und Ansätze aus beiden Bereichen, so findet man eine gemeinsame Menge an Unzulänglichkeiten, die einer benutzerfreundlichen und gleichzeitig gut wartbaren Lösung entgegen stehen:

- In den meisten Fällen wird Detailwissen über die Implementierung der integrierten Applikationen benötigt, eine Abstraktionsschicht fehlt in der Regel. Dies macht die Integration zeitaufwändig und führt zu einer *engen Kopplung*, die das Gesamtsystem bei Änderungen der integrierten Applikationen potentiell instabil macht.
- Ein gemeinsames Eventmodell fehlt ebenso wie Möglichkeiten der Konvertierung zwischen Datenmodellen der integrierten Applikationen, so dass der Nachrichtenaustausch sowie der *Abgleich konzeptionell heterogener Datenmodelle* größtenteils manuell implementiert werden muss.

- *Technologisch heterogene Applikationen* lassen sich in der Regel nicht beliebig integrieren.

Besondere Probleme bereiten hierbei Interaktionen zwischen Applikationen: soll etwa eine Applikation auf Auswahlaktionen in einer anderen Applikation durch Anzeige von Details reagieren, oder Ziehen und Fallenlassen von Objekten aus einer Applikation in eine andere ermöglicht werden, so treten diese Nachteile besonders auffällig in Erscheinung. In der Regel müssen solche Interaktionen manuell unter Bezugnahme auf Wissen über die Implementierung der Applikationen codiert werden, was Abhängigkeiten zwischen den Applikationen schafft und zu dem Paradigma der losen Kopplung im Widerspruch steht.

Vielen dieser Unzulänglichkeiten lässt sich durch eine zusätzliche Abstraktionsschicht begegnen. In dieser Arbeit werden Ontologien genutzt, um eine solche Abstraktionsschicht zu definieren.

2 Ontologien in der Applikationsintegration

Ontologien sind „formale Spezifikationen einer Konzeptualisierung“ [Gru95], also ein formales Modell der Begriffe, die eine Domäne beschreiben, und ihrer Zusammenhänge. Da sie das Wissen über eine Domäne formal codieren, können auch komplexe Abfragen, die die Kombination verschiedener Fakten zur Beantwortung benötigen, automatisch mit Hilfe von automatischem Schlussfolgern, dem sogenannten *Reasoning*, beantworten werden. Der Nutzen von Ontologien für die Systemintegration wurde bereits früh erkannt; so wurde bereits in einem häufig zitierten Artikel von Uschold und Grüninger aus dem Jahr 1996 die potentielle Verwendung von Ontologien als *inter-lingua* zwischen Systemen zur Herstellung von Interoperabilität vorgeschlagen [UG96].

Die ersten Arbeiten zur Integration auf Datenhaltungsebene mit Ontologien gehen auf die 1980er Jahre zurück. In solchen Arbeiten werden Ontologien genutzt, um eine Abstraktionsschicht über verschiedenen Datenbanken zu bilden. Komplexe Anfragen können mit Hilfe von Begriffen der Ontologie gestellt werden, und ein Integrationssystem übersetzt die Anfragen an die zugrundeliegenden Datenbanksysteme, kombiniert die zurückgelieferten Datensätze, leitet ggf. weitere Fakten daraus ab und liefert eine konsolidierte Antwort zurück [DH05].

Auf der Ebene der Applikationslogik werden Ontologien hauptsächlich im Bereich der *Semantic Web Services* eingesetzt. Hier werden klassische Web-Service-Beschreibungen um formale semantische Beschreibungen angereichert. Damit wird im Idealfall ein komplette, maschinenverständliche Beschreibung eines Dienstes inklusive seiner Vor- und Nachbedingungen sowie seiner Ausführungssemantik und seiner nichtfunktionalen Eigenschaften zur Verfügung gestellt, um so eine automatische Kombination und Ersetzung von Diensten zu ermöglichen [SGA07].

Im Gegensatz zu diesen Ansätzen existieren bislang keine Ansätze, die Ontologien für die Applikationsintegration auf Nutzerschnittstellenebene vorsehen.

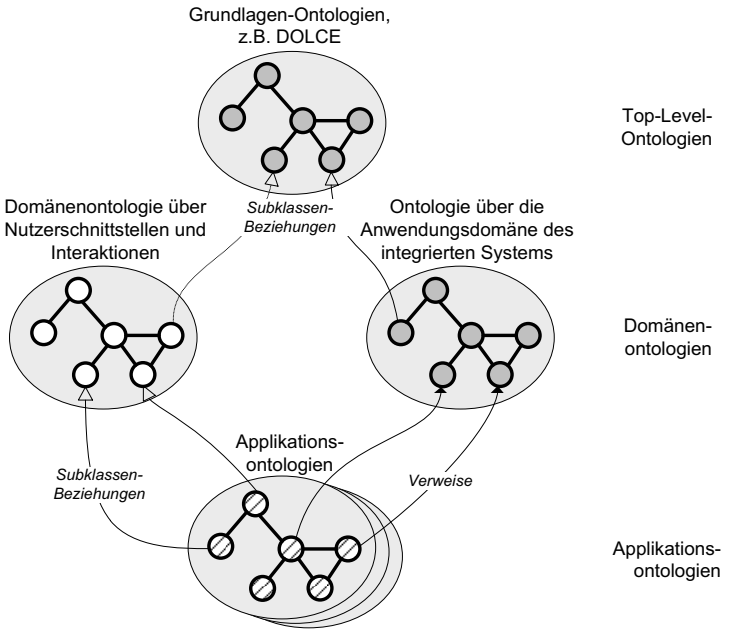

Abbildung 2: Eingesetzte Ontologien, dem Schichtenmodell von Guarino [Gua98] folgend

3 Ansatz

Um eine möglichst *lose Kopplung* der integrierten Applikationen zu erreichen, werden Ontologien überall dort eingesetzt, wo ein Informationsaustausch zwischen den Applikationen stattfinden muss. Sämtliche zwischen den Applikationen ausgetauschten Nachrichten werden komplett mit Hilfe von Ontologien beschrieben, was auch die involvierten Geschäftsobjekte einschließt. Auch Statusinformationen der einzelnen Applikationen, die für andere Applikationen relevant sein können, werden ausschließlich mit Hilfe von Ontologien codiert. Insgesamt kommen drei verschiedene Arten von Ontologien zum Einsatz, wie in Abb. 2 gezeigt:

- Jede einzelne Applikation wird in ihrer eigenen Applikationsontologie beschrieben. Diese formalisiert sämtliche Interaktionsmöglichkeiten mit dieser Applikation. Die Applikationsontologie baut auf zwei verschiedenen Ontologien auf.
- Eine Domänenontologie über Nutzerschnittstellen und Interaktionen beschreibt auf abstrakter Ebene, aus welchen Arten von Komponenten und Interaktionen eine Nutzerschnittstelle bestehen kann.
- Eine Domänenontologie der Geschäftsdomäne beschreibt die konkrete Anwendungsdomäne des integrieten Systems, z.B. das Bankenwesen.

Alle Ontologien werden nutzen die gemeinsame Top-Level-Ontologie DOLCE [MBG+03], wobei dies für die Domänenontologie der Geschäftsdomäne fakultativ ist. Letztere ist nicht Teil des in der Arbeit entwickelten Ansatzes, der domänenunabängig ist.

Jede integrierte Applikation wird in einen sogenannten *Container* verpackt, der die Funktionalität der Anwendung nach außen kapselt (siehe Abb. 3). Der Container stellt die Daten der Anwendungen als *Linked Data* [BHBL09] zur Verfügung, so dass ein Reasoner darauf zugreifen kann. Hierzu werden die Daten der Anwendung mit Hilfe von *Transformationsregeln* in ein ontologiebasiertes Format überführt. Weiterhin übernimmt der Container die Kommunikation mit anderen Anwendungen über Events. Hierzu gehört insbesondere die Koordination von applikationsübergreifenden Interaktionen, z.B. Drag-and-Drop-Aktionen.

Jede Nachricht und jedes Datenobjekt, das zwischen Applikationen ausgetauscht wird, ist komplett mit Ontologien beschrieben. So kann in einer Kombination der verschiedenen Ontologien zum Beispiel ausgedrückt werden, dass der Nutzer eine Aktion vom Typ *Auswählen* mit einem Objekt vom Typ *Kunde* durchgeführt hat, wobei dieses Objekt wiederum weitere Eigenschaften haben kann. Die Ontologie als *inter-lingua* ermöglicht jeder Applikation, Nachrichten von anderen Applikationen zu verstehen, ohne die zugrunde liegende technische Implementierung zu kennen. Damit verlässt sich jede Anwendung lediglich darauf, Nachrichten mit bestimmten Annotationen zu erhalten, direkte Abhängigkeiten zwischen Applikationen werden so eliminiert.

Um die Entkopplung zwischen Applikationen noch stärker voranzutreiben und damit die Modularität und Wartbarkeit des resultierenden Systems noch weiter zu erhöhen, reagieren Container nicht direkt auf Events aus anderen Containern. Die Koordination von anwendungsübergreifenden Interaktionen wird mit Hilfe von *Interaktionsregeln* implementiert, die von einem Reasoner zentral verarbeitet werden. Dieser Reasoner hat Zugriff auf sämtliche Daten und Applikationsontologien. Da die Interaktionsregeln nur auf Konzepten der Ontologien auf Domänenebene, aber nicht auf Konzepten der einzelnen Applikationsontologien formuliert werden, müssen sie in der Regel nicht angepasst werden, wenn sich die Konfiguration des integrierten Gesamtsystems ändert, so dass eine größtmögliche Entkopplung gewährleistet ist.

Der Reasoner kann außerdem auch komplexe Sachverhalte, die in den Domänenontologien modelliert sind, berücksichtigen. So kann beispielsweise folgende Interaktionsregel definiert werden: *Wenn in der Stammdatenanwendung ein Kunde mit Premiumstatus ausgewählt wurde, markiere in der Produktübersicht alle Produkte für Premiumkunden*. Welche Fakten und Axiome dazu führen, dass ein Kunde Premiumstatus bekommt, kann dabei in der Domänenontologie modelliert sein – es lässt sich also zusätzlich zur Entkopplung bei der Integration auch Domänenwissen aus den Anwendungen herausfaktorisieren und anwendungsübergreifend nutzbar machen.

Der Einsatz von Reasoning zur Verarbeitung der Interaktionsregeln ermöglicht zudem eine sehr abstrakte Form der Interaktionsbeschreibung. So ist es zum Beispiel möglich, eine Landkarten-Anwendung wie *Google Maps* mit einer wie folgt formulierten Interaktionsregel zu integrieren: *Wenn der Nutzer ein Objekt, das eine Position hat, auswählt, wird diese Position auf der Karte markiert*. Die Bestimmung, ob ein konkretes Objekt in diese

Kategorie gehört, übernimmt der Reasoner, und es muss nicht a priori bekannt sein, *welche* Arten von Objekten mit Positionen später vom System verarbeitet werden. Damit können Applikationen zum integrierten System hinzugefügt werden und interagieren automatisch mit bereits existierenden Applikationen, ohne dass weitere Anpassungen nötig werden.

Auch globale Interaktionsregeln können definiert werden, um zum Beispiel ein *Linked-Views-Paradigma* zu implementieren: *Wenn der Nutzer ein Objekt selektiert, das auch in anderen Anwendungen bekannt ist, dann hebe dieses Objekt in allen Anwendungen hervor*. Diese Regel kommt ohne Referenz auf die Domänenontologie aus und beschreibt daher eine domänenübergreifende Interaktion, die automatisch greift, sobald zwei Applikationen dieselbe Art von Objekten verarbeiten und die entsprechenden Annotationen an ihre Nachrichten anfügen. Mit Hilfe solcher globaler Interaktionsregeln kann eine grundlegende ad-hoc-Interoperabilität zwischen Anwendungen ermöglicht werden, die greift, ohne dass spezielle Regeln für einzelne Anwendungen definiert werden müssen.

4 Zentrale wissenschaftliche Beiträge

Ein zentrales Artefakt der Arbeit ist die formale Ontologie über Nutzerschnittstellen und Interaktionen [PP11]. Sie stellt ein abstraktes Modell von Nutzerschnittstellen dar, das eine große Bandbreite von Interaktionsmodalitäten abdeckt und zudem durch Nutzung der Top-Level-Ontologie DOLCE stark formalisiert ist. Mit rund 200 Klassen und 500 formalen Axiomen ist sie die derzeit ausdrucksstärkste und umfassendste Ontologie dieser Domäne. Gemeinsam mit formalen Regeln fungiert diese Ontologie als Abstraktionsschicht zwischen integrierten Applikationen und entkoppelt diese voneinander. Da nur noch jede integrierte Applikation auf die Ontologie abgebildet werden muss, sinkt die Komplexität der Integration von $O(n^2)$ auf $O(n)$, und die schwache Kopplung gewährleistet die Wartbarkeit der Anwendung. Eine solche Ontologie kann auch in anderen Bereichen, wie zum Beispiel dem Austausch und der Konvertierung von Nutzerschnittstellenmodellen in der modellgetriebenen Entwicklung, sinnvoll eingesetzt werden.

Durch die Einführung eines abstrakten Modells ist es möglich, sowohl konzeptionell als auch technologisch heterogene Applikationen miteinander zu koppeln. Obgleich es bereits Ansätze gibt, Daten aus Applikationen in ontologiebasierter Form darzustellen, verlassen sich diese in der Regel darauf, dass die Datenmodelle der Applikationen strukturell ähnlich zu der Ontologie sind – eine Annahme, die in der Praxis kaum eintrifft. In dieser Arbeit wurde diesen Ansätzen ein weitaus flexiblerer, regelbasierter Ansatz gegenübergestellt, der in der Lage ist, auch konzeptionell unterschiedliche Klassenmodelle auf beliebige Ontologien abzubilden [POPP12]. Dieser Ansatz ermöglicht nicht nur den Objektaustausch, sondern generell die Entwicklung semantischer Programmiermodelle in weitaus flexiblerer Form, als das bisherige Ansätze leisten.

Neben der konzeptionellen Heterogenität unterstützen die verfügbaren Ansätze zur Integration auf Nutzerschnittstellenebene die Überbrückung technologischer Heterogenität nur unzulänglich. Insbesondere die Implementierung nahtloser Interaktionen, wie z.B. Drag and Drop zwischen Anwendungen, ist bei technologisch verschiedenen Nutzerschnitt-

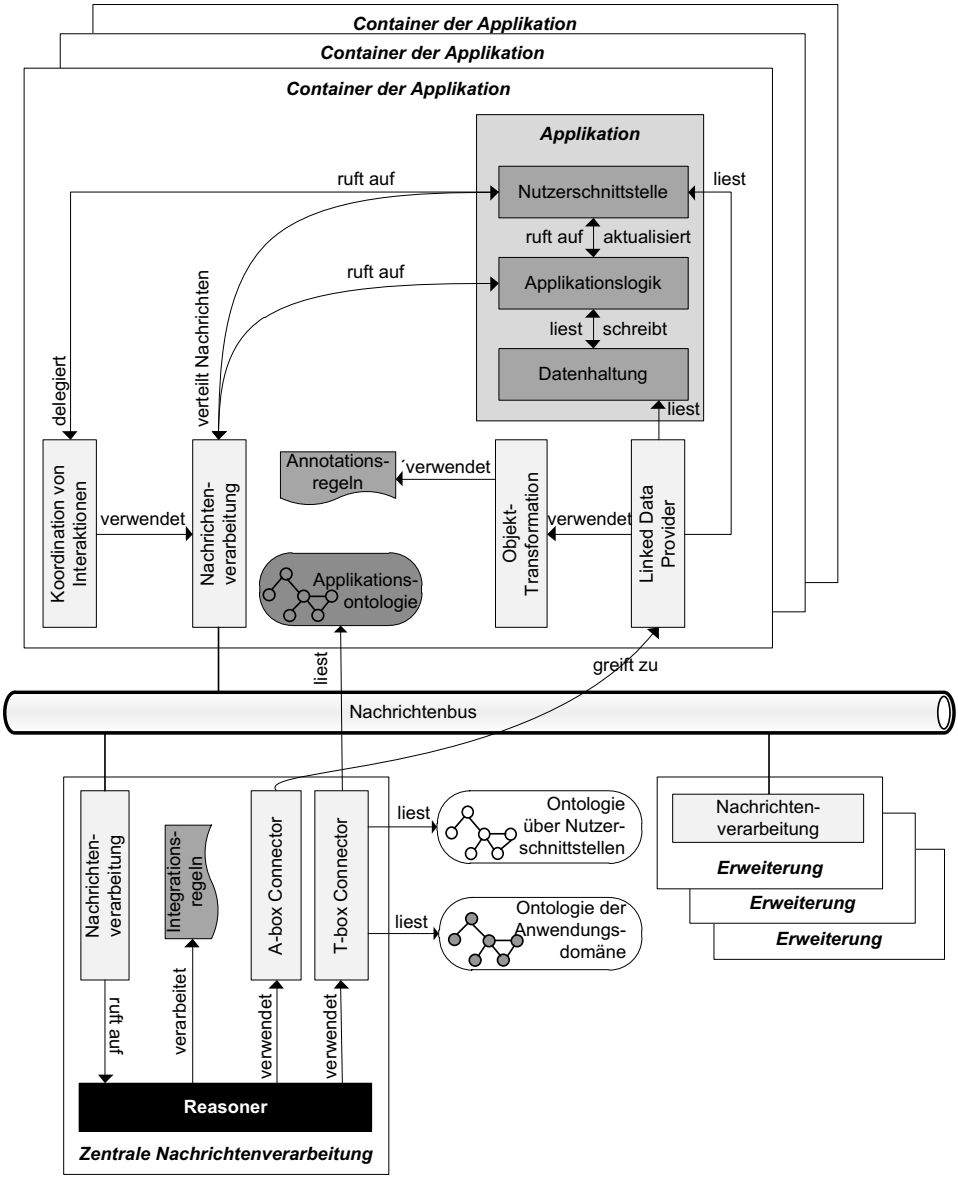

Abbildung 3: Überblick über die Architektur [Pau11b]

stellen, wenn überhaupt, nur sehr aufwändig machbar. Im Rahmen dieser Arbeit wurde am Beispiel von Flex- und Java-Komponenten gezeigt, dass die Einführung einer abstrakten Zwischenschicht auch solche Interaktionen direkt unterstützt [PE10].

Im Umgang mit Nutzerschnittstellen ist die Reaktivität des Systems entscheidend. Gerade bei größeren Faktenmengen stellt die semantische Nachrichtenverarbeitung, wie sie in dieser Arbeit eingesetzt wird, oftmals ein Problem dar, da die Reaktionszeiten für interaktive Systeme deutlich zu lang werden. Im Rahmen dieser Arbeit wurden verschiedene Architekturansätze gegenübergestellt und analysiert. Mit der gewählten Referenzimplementierung ist es möglich, die Reaktionszeiten auch für eine große Anzahl von Applikationen und eine hohe Nachrichtenfrequenz noch deutlich unter der Grenze von einer halben Sekunde zu halten, um eine nutzerfreundliche Interaktion zu gewährleisten [Pau10]. Da die Reaktionszeit in der Praxis oftmals ein Hinderungsgrund für den Einsatz von semantischen Technologien ist, zeigt die Referenzimplementierung Wege auf, wie ontologiegestützte Anwendungen in performanter Form entwickelt werden können, und ebnet damit den Weg für bisher aus Performancegründen nicht umsetzbare Applikationen.

Die Integration mit Hilfe von semantisch annotierten Daten ermöglicht auch neue Interaktionsformen. Da bei dem in der Anwendung gewählten Ansatz sämtliche Daten als Linked Data zur Verfügung stehen, ist es möglich, auf diese neben den existierenden Nutzerschnittstellen parallel mit einem Linked Data Browser zuzugreifen und die Daten zusätzlich als semantisches Netz zu visualisieren. Im Rahmen der Arbeit konnte in einer Nutzerstudie in der Domäne des Katastrophenschutzes gezeigt werden, dass in dieser Domäne typischerweise anfallende Aufgaben mit einer solchen Visualisierung schneller und mit größerer Nutzerzufriedenheit gelöst werden können [Pau11a].

Das in dieser Arbeit entwickelte Framework zur Applikationsintegration auf Ebene der Nutzerschnittstellen wurde im Projekt *SoKNOS* eingesetzt, um ein IT-Unterstützungssystem für Krisenstäbe im Katastrophenschutz zu entwickeln [PDTS+09]. Das System integriert unterschiedlichste Module von der Ressourcenverwaltung über die Nachrichtenverarbeitung bis hin zu interaktiven Lagekarten. Mit Hilfe des Frameworks konnten hier insgesamt 20 verschiedene Anwendungen mit ca. 180 verschiedenen applikationsübergreifenden Interaktionsformen integriert werden. Das System *SoKNOS*, das auf Basis des in dieser Dissertationen entwickelten Frameworks implementiert wurde, wurde bei verschiedenen Messen vorgeführt und mit Endanwendern aus der Katastrpohenschutzdomäne getestet. Die Erfahrungen aus dem Projekt *SoKNOS* zeigen, dass das in dieser Arbeit entwickelte Framework damit auch Anforderungen realer, komplexer IT-Systeme genügt.

5 Zusammenfassung und offene Forschungsfragen

Diese Arbeit stellt einen neuartigen Ansatz vor, um Anwendungsintegration auf Ebene der Nutzerschnittstellen zu ermöglichen. Mit Hilfe von Ontologien werden applikationsübergreifende Interaktionen ermöglicht, ohne das Paradigma der schwachen Kopplung aufzugeben. Der Ansatz erlaubt die Überbrückung von technologischen und konzeptionellen Heterogenitäten und gewährleistet die schnelle Verarbeitung semantisch annotierter Nach-

richten, um den Interaktionsfluss nicht zu verlangsamen. Zudem konnten in einer Nutzerstudie Verbesserungen in der Bedienbarkeit integrierter Systeme durch Visualisierung semantisch annotierter Daten nachgewiesen werden. Zahlreiche Teilergebnisse dieser Arbeit haben Einfluss auf verwandte Forschungsgebiete, in denen Ontologien und semantische Technologien zum Einsatz kommen.

Der in dieser Arbeit entwickelte Ansatz eröffnet weitere relevante Forschungsfragen. So konzentriert sich die Arbeit allein auf Interaktionen mit einem Gerät und einem Endbenutzer; Interaktionen mit multi-modalen Nutzerschnittstellen wurden ebenso ausgeklammert wie Mehrbenutzerinteraktionen. Während die entwickelte Ontologie diese Interaktionsformen prinzipiell abbilden kann, ist die praktische Umsetzung ebenso wie die Benutzbarkeit solcher integrierten Systeme zu untersuchen.

Die in der Arbeit entwickelten Ontologien bilden die integrierten Anwendungen und ihre Interaktionen formal ab. Dies würde es prinzipiell ermöglichen, diese formalen Beschreibungen nicht nur der Maschine, sondern in geeigneter Form auch dem Menschen zugänglich zu machen, um eine explorative Erfahrung des integrierten Systems zu ermöglichen. Geeignete Interaktionsparadigmen und Visualisierungsformen wären hier zu erforschen.

Literatur

[BHBL09] Christian Bizer, Tom Heath und Tim Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009.

[DH05] AnHai Doan und Alon Y. Halevy. Semantic Integration Research in the Database Community: A Brief Survey. *AI Magazine*, 26(1):83–94, 2005.

[DYB+07] Florian Daniel, Jin Yu, Boualem Benatallah, Fabio Casati, Maristella Matera und Regis Saint-Paul. Understanding UI Integration: A Survey of Problems, Technologies, and Opportunities. *IEEE Internet Computing*, 11(3):59–66, 2007.

[Fow03] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison Wesley, 2003.

[Gru95] Thomas R. Gruber. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal Human-Computer Studies*, 43(5-6):907–928, 1995.

[Gua98] Nicola Guarino, Hrsg. *Formal Ontology and Information Systems*. IOS Press, 1998.

[MBG+03] Claudio Masolo, Stefano Borgo, Aldo Gangemi, Nicola Guarino und Alessandro Oltramari. WonderWeb Deliverable D18 – Ontology Library (final). `http://wonderweb.semanticweb.org/deliverables/documents/D18.pdf`, 2003. Letzter Zugriff: 17.04.2012.

[MR92] Brad A. Myers und Mary Beth Rosson. Survey on user interface programming. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems*, Seiten 195–202. ACM, 1992.

[Pau10] Heiko Paulheim. Efficient Semantic Event Processing: Lessons Learned in User Interface Integration. In Lora Aroyo, Grigoris Antoniou, Eero Hyvönen, Annette ten Teije, Heiner Stuckenschmidt, Liliana Cabral und Tania Tudorache, Hrsg., *The Semantic Web: Research and Applications (ESWC 2010), Part II*, number 6089 in LNCS, Seiten 60–74. Springer, 2010.

[Pau11a] Heiko Paulheim. Improving the Usability of Integrated Applications by Using Interactive Visualizations of Linked Data. *International Journal on Computer Science and Applications (IJCSA)*, 8(2), 2011.

[Pau11b] Heiko Paulheim. *Ontology-based Application Integration*. Springer, 2011.

[PDTS+09] Heiko Paulheim, Sebastian Döweling, Karen Tso-Sutter, Florian Probst und Thomas Ziegert. Improving Usability of Integrated Emergency Response Systems: The SoKNOS Approach. In *Proceedings der 39. Jahrestagung der Gesellschaft für Informatik e.V. (GI) – Informatik 2009*, number 154 in LNI, Seiten 1435–1449, 2009.

[PE10] Heiko Paulheim und Atila Erdogan. Seamless Integration of Heterogeneous UI Components. In Noi Sukaviriya, Jean Vanderdonckt und Michael Harrison, Hrsg., *Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS 2010)*, Seiten 303–308. ACM, 2010.

[POPP12] Heiko Paulheim, Daniel Oberle, Roland Plendl und Florian Probst. An Architecture for Information Exchange Based on Reference Models. In *4th International Conference on Software Language Engineering (SLE)*, number 6940 in LNCS. Springer, 2012.

[PP10] Heiko Paulheim und Florian Probst. Application Integration on the User Interface Level: an Ontology-Based Approach. *Data & Knowledge Engineering Journal*, 69(11):1103–1116, 2010.

[PP11] Heiko Paulheim und Florian Probst. A Formal Ontology on User Interfaces – Yet Another User Interface Description Language? In Tim Hussein, Stephan Lukosch, Heiko Paulheim, Jürgen Ziegler und Gaelle Calvary, Hrsg., *Proceedings of the Second Workshop on Semantic Models for Adaptive Interactive Systems (SEMAIS)*, 2011.

[SGA07] Rudi Studer, Stephan Grimm und Andreas Abecker, Hrsg. *Semantic Web Services - Concepts, Technologies and Applications*. Springer, 2007.

[UG96] Mike Uschold und Michael Grüninger. Ontologies: Principles, Methods and Applications. *Knowledge Engineering Review*, 11:93–136, 1996.

Heiko Paulheim wurde am 1.6.1977 in Kassel geboren. Er studierte Mathematik und Anglistik an der Universität Kassel sowie Informatik an der Hochschule Darmstadt und an der Technischen Universität Darmstadt. Er forschte an der Hochschule Darmstadt über den Einsatz von Ontologien im elektronischen Geschäftsverkehr und promovierte bei SAP Research und an der Technischen Universität Darmstadt über die ontologie-basierte Applikationsintegration auf Nutzerschnittstellenebene. Seit 2011 lehrt und forscht er an der Technischen Universität Darmstadt im Fachgebiet Knowledge Engineering, seine aktuellen Forschungsfelder umfassen u.a. die Anwendung von Machine-Learning-Verfahren auf Linked Open Data, Ontology Learning und Ontology Matching. Heiko Paulheim ist Mitglied in zahlreichen Programm-Komitees im Bereich Semantic Web und Künstliche Intelligenz sowie Mitausrichter der Workshopreihen *SEMAIS* (Semantic Models for Adaptive Interactive Systems) und *Know@LOD* (Knowledge Discovery and Data Mining Meets Linked Open Data).