Technical Report

Nr.: FIN-03-2008

Die Sprachbestandteile von Domänenspezifischen Sprachen:

Eine Ableitung aus den sprachphilosophischen und linguistischen Wurzeln der Informatik

Sebastian Günther

Arbeitsgruppe Wirtschaftsinformatik

Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg

Impressum (§ 10 MDStV):

Herausgeber:
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Der Dekan

Verantwortlich für diese Ausgabe:
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Sebastian Günther
Postfach 4120
39016 Magdeburg
E-Mail: sebastian.guenther@ovgu.de

http://www.cs.uni-magdeburg.de/Preprints.html

Auflage: 51

Redaktionsschluss:

Herstellung: Dezernat Allgemeine Angelegenheiten,
Sachgebiet Reproduktion

Bezug: Universitätsbibliothek/Hochschulschriften- und
Tauschstelle

Die Sprachdefinitionsbestandteile von Domänenspezifischen Sprachen: Eine Ableitung aus den sprachphilosophischen und linguistischen Wurzeln der Informatik

Sebastian Günther

Abstract: Die Informatik ist ein vielfältiges Forschungsgebiet. Ein Kerngebiet ist die Forschung an Sprachen. In der Informatik werden Sprachen benutzt, um den Aufbau von Computern und Software abzubilden, um große Datenbestände zu formalisieren und Programmabläufe zu beschreiben. Sprachforschung hat sich aus der Sprachphilosophie und der Linguistik heraus entwickelt. Diese Wurzeln scheinen in der Informatik in Vergessenheit zu geraten, zeigt sich doch ein Definitionschaos bei den zentralen Begriffen Syntax, Semantik und Pragmatik. Dies ist auch bei den Domänenspezifischen Sprachen - speziellen Programmiersprachen - der Fall. Solche Sprachen sind sehr vielversprechend, die Probleme im Software Engineering lösen zu können. Dazu muss jedoch eine einheitliche Definitions- und Begriffsbasis gebaut werden. Dieser Beitrag rekonstruiert die sprachphilosophischen und linguistischen Wurzeln der Sprachdefinitionsbestandteile in der Informatik, um eine stabile Grundlage für weitere Forschungen an Domänenspezifischen Sprachen zu schaffen.

Inhaltsverzeichnis

1 Einleitung

1.1 Die Probleme in der Softwareentwicklung

Die Entwicklung von Software ist ein Kerngebiet der Informatik. Software ist heute ein kritischer Faktor in vielen Unternehmen. Ohne Software sind innovative Produkte und Dienstleistungen nicht mehr möglich [BJNR06, S. 210]. Trotz einer nunmehr 40-jährigen Geschichte ist Softwareentwicklung immer noch mit vielen Schwierigkeiten verbunden, die nachfolgend aufgeführt werden.

Die Erstellung von Software erfordert von den Entwicklern ein enormes Wissen sowohl in der Anwendungsdomäne als auch in der Umsetzung. Im Prinzip werden die natürlichsprachlichen Beschreibungen in Programmcode umgesetzt - Programmieren ist nichts anderes als das Umsetzen der Informationen einer sprachlichen Realisierung in eine andere. Von bedeutenden Ausnahmen abgesehen, haben diese Umsetzungen den Charakter einer "Einmalentwicklung".

Software besitzt auch eine immense Komplexität, um die vielfältigen Anforderungen umzusetzen. Vielfältiges Wissen fließt in die Softwareentwicklung ein: Konzepte der Anwendungsdomäne, Systemkonfiguration und Deployment sowie Patterns [GSCK04, S. 120ff.]. Dieses Wissen und die Gesamtfunktionalität eines Systems wird jedoch über verschiedene Entwicklungsartefakte[1] verstreut [GSCK04, S.37]. Die Streuung geschieht sogar noch über verschiedene Programmiersprachen hinweg. Moderne Webanwendungen sind in einem Sammelsurium unterschiedlichster Sprachen entwickelt: SQL für die Datenhaltung, PHP/Ruby/Java für die Anwendungslogik, HTML und CSS für die Darstellung und JavaScript für die Client-seitige Funktionalität.

Durch die manuelle Erstellung von Software entstehen vielfältige Probleme. Die nachträgliche Änderung des Systems wird erschwert und ist mit hohen Kosten verbunden [PT07, S. 162]. Ebenso entstehen durch den weitgehend manuellen Transformationsprozess der Anforderungen viele Probleme, wie Performance Schwierigkeiten, Fehler in der Funktionalität, nicht verstandene oder nicht gelöste Anforderungen und weiteres [GSCK04, S. 47ff.]. Diese Missstände resultieren in hohen Fehlerraten von Softwareprojekten und sind unter anderem verantwortlich für die hohe Entwicklungskomplexität [Sta95].

Diese Probleme sind in der Informatik nicht neu, aber noch immer aktuell. Verschiedene Ansätze wollen die aufgezeigte Komplexität einschränken. Die generative Programmierung [CE00, S.137] und die Software-Fabriken [GSCK04, S. 142f.] erstellen durch Modellierungssprachen und Programmgeneratoren wiederverwendbare Software System-Familien. Die Aspekt-Orientierte Programmierung [KLM+97] und die Feature-Orientierte Programmierung [Pre97] versuchen, einzelne Funktionsaspekte eines Systems zu kapseln und in wiederverwendbaren Quellcode auszudrücken. Auch die Domänenspezifischen Sprachen (DSL) können diese Probleme lösen.

[1] Spezifikationen, Dokumentation, Quellcode und Maschinencode sind künstlich geschaffene Objekte (Artefakte) in der Softwareentwicklung.

1.2 Domänenspezifische Sprachen als Lösung

DSL bezeichnen eigenständige und komprimierte Programmiersprachen, die auf ein eigenständiges Anwendungsgebiet ausgerichtet sind [Hud98, S. 134]. Sie erfassen und beschreiben die Konzepte (verstanden als Begriffe) und ihre Interaktionen in einem sprachlichen Modell, dass zugleich auch Quellcode ist. Fachsprache (beinhaltet eine Reihe spezialisierter, in Beziehung zueinander stehender Begriffe) und Programmiersprache (vom Computer interpretier- und ausführbar) verschmelzen in einer DSL miteinander. Sie lösen die geschilderten Softwareentwicklungsprobleme auf folgende Weise. *Erstens* wird die Einmalentwicklung verhindert, weil die Domänenkonzepte wiederverwendet werden können. *Zweitens* ist das Entwicklungs- und Domänenwissen in den DSL enthalten und wird nicht zerstreut. Und *drittens* kann aus der DSL heraus die Anwendung automatisch generiert werden. Damit haben DSL ein großes Potential, die Probleme des Software Engineering zu lösen.

Allerdings gibt es einige Defizite in diesem Forschungsgebiet. Dazu gehören unter anderem eine divergente Terminologie und die unterschiedliche Auffassungen darüber, wie eine DSL als Sprache beschrieben wird. Nach der Ansicht von Forschern aus der Informatik und der Linguistik wird Sprache durch die Syntax (grammatikalische Form), die Semantik (Bedeutung) und die Pragmatik (weitere Bedeutungsaspekte) definiert [Kut75, S. 17ff.],[SR89, S. 299] [GSCK04, S. 280f.]. Diese zentralen Begriffe werden als *Sprachdefinitionsbestandteile*, das heißt verschiedene Gesichtspunkte bei der Beschreibung von Sprachen, bezeichnet. Wie verschiedentlich die Ansicht über die Sprachdefinitionsbestandteile in der Literatur zu DSL ist, zeigen die folgenden, rein deskriptiven, Beispiele:

- Semantik als Transformation in Java-Klassen [BCDVM02, S. 13] oder als Konzepte eines Programms [Bor95, S. 671].
- Semantik wird unterschieden in “Translational Semantic” (Kompilierungszeitpunkt) und “Trace-Based Semantic” (Interpretationszeitpunkt) [GSCK04, S. 310 ff.]
- Syntax-Beschreibung durch eine Backus-Naur-Form und ein UML-basiertes Meta-Modell [GSCK04, S. 286 ff.]
- Syntax-Beschreibung durch Description Logics [Bor95, S. 675] oder Orientierung an gewöhnliche Programmiersprachen, ohne diese explizit zu nennen [VDBHKO02, S. 338f.].

Diese Ansichten lassen darauf schließen, dass die Sprachdefinitionsbestandteile mehrdimensionale Begriffe sind und mit unterschiedlichen Verfahren beschrieben werden können. Auch für andere Bereiche lassen sich solche Beispiele finden. Um zur Klärung der aufgezeigten Probleme zu gelangen, werden die folgenden fünf Forschungsfelder für DSL identifiziert.

1. DSL-Begriff und Taxonomie von DSL-Arten
2. Erarbeitung der Sprachdefinitionsbestandteile
3. Systematik der Sprachbeschreibungsverfahren
4. Konstruktionsprozesse für DSL
5. DSL im Softwareengineering

1.3 Ziel und Vorgehen des Berichtes

Dieser Bericht widmet sich dem zweiten Forschungsfeld und erarbeitet die Sprachdefinitionsbestandteile von DSL. Um zu einer grundlegenden und die historischen Wurzeln berücksichtigenden Beschreibung zu gelangen, beginnt die Untersuchung bei der Sprachphilosophie. Dass die Sprachphilosophie relevant ist, zeigt eine Reihe von Veröffentlichungen. Betrachtungen zum Charakter von Sprache und Informatik sind z. B. in [Ort02] und [Ort05] zu finden. Der Einfluss von linguistischen Theorien auf die Forschung an Informationssystemen wird in [Dre06] gezeigt. Und aktuelle Werke wie [FN08] und [Yet08] behandeln explizit die Einführung sprachphilosophischer und linguistischer Theorien in die Wirtschaftsinformatik. Bis heute profitiert die Informatik von sprachphilosophischen Grundlagen. In [WH07] wird empfohlen, zur Klärung der wissenschaftlichen Forschungsmethodik beizutragen, indem die verwendeten Methoden angegeben werden. Dieser Bericht verwendet eine umfangreiche *Literaturanalyse*, um die relevanten Begriffe der Sprachbestandteile und ihre Bedeutung zu erarbeiten. Diese Erkenntnisse werden durch *argumentativ-deduktive* Methoden weiter untersucht und zueinander in Beziehung gesetzt.

Vor diesem Hintergrund wird die Arbeit folgendermaßen gegliedert. Nach der Einleitung wird im zweiten Kapitel ein Überblick zur Sprachphilosophie und den Forschungsgebieten dieser Disziplin erarbeitet. Im dritten Kapitel werden die zentralen Sprachbestandteile für die Linguistik und die Informatik erarbeitet und zueinander in Beziehung gesetzt. Die Kombination dieser Erkenntnisse fließt dann im vierten Kapitel zusammen, um die Sprachbestandteile von DSL zu definieren. Das fünfte Kapitel fasst den Forschungsbericht zusammen und gibt einen Ausblick auf weitere Arbeiten.

2 Sprache & Sprachphilosophie

Sprachphilosophie bezeichnet ein Gebiet der Sprachforschung, dessen Fragestellungen sich bis in die Antike zurückverfolgen lassen. In diesem Abschnitt wird ein komprimierter Überblick zu diesem sehr weitläufigen Feld gegeben. Nach ein paar einleitenden Worten über die Sprache an sich wird das Wesen der Sprachphilosophie definiert. Daran schließt sich die Aufzählung der Forschungsgebiete der Sprachphilosophie und die Unterscheidung

zu den Gebieten der Linguistik an. Eine Zusammenfassung schließt dieses Kapitel.

2.1 Über die Natur der Sprache

Kein Phänomen dieser Welt verfügt über ähnliche Macht und Eigenschaften wie die Sprache. Noam Chomsky bezeichnet den Sprachgebrauch als äußerst "produktiv und potentiell unendlich in seiner Reichweite" und zeigt damit das enorme kreative Potential von Sprache und ihren Sprechern auf [Cho92, S. 27f.]. Die Funktionen von Sprache sind vielfältig: "wir fragen, befehlen, behaupten, [...] begrüßen, begründen, werten" [Kut75, S. 25] - kurzum: Wir tauschen uns aus und teilen uns mit. Denken und Sprache sind fest miteinander verknüpft [Who63, S. 77]. Sprache ist als "reflection on reality" [Mar96, S. 4] anzusehen, also als das bewusste Interpretieren und Wiedergeben der beobachteten Umwelt. Andere Autoren bezeichnen Sprache auch als Verständnissystem oder Fähigkeit ihrer Anwender sich auszudrücken [SR89, S. 316]. Diese Eigenschaften machen Sprache zu einem Universalgegenstand, um Wissen zu repräsentieren und auch zu gewinnen. Diese Mächtigkeit der Sprachen und die Vielfalt der Möglichkeiten spiegeln auch die Begriffe der Informatik wider, wenn Begriffe wie Programmierungssprachen, Modellierungssprachen und Auszeichnungssprachen verwendet werden.

2.2 Was ist Sprachphilosophie?

Aufgrund ihrer Eigenschaften ist Sprache ein Phänomen, welches bereits von ersten Hochkulturen zu einem Forschungsgegenstand erhoben wurde. So beschäftigten sich bereits in der Antike die Philosophen Platon und Aristoteles mit sprachphilosophischen Fragen [Ber99, S. 25ff. und S. 43ff]. Die Sprachphilosophie entwickelte sich aus der Ontologie heraus; sie stammt von den Fragestellungen über das Sein, Logik und Wahrheit ab [Ber99, S. 15].

Verschiedene Ansichten darüber, was Sprachphilosophie ist, sind seit der Antike entstanden. Zur Definition hilft ein Rückbezug auf den Begriff der Philosophie an sich. Was genau ist Philosophie und worin unterscheidet sie sich von einer Wissenschaft? Die Erläuterung dieser Frage kann dadurch geschehen, dass nach der Art der Fragestellung und damit nach dem Ziel von Forschung unterschieden wird. Coseriu unterscheidet dabei erstens die Fragen nach einem Einzelgegenstand, zweitens die Frage nach den Eigenschaften eines Gegenstandes, und drittens nach dem Sinn der Existenz und dem Warum von Gegenständen [Cos03, S. 1ff.]. Die dritte Fragestellung ist in ihrer Natur philosophisch. Wenn Gegenstände nicht als Objekte, sondern als beliebige Klassen und damit auch Begriffe verstanden werden, dann ist die Aufgabe einer Sprach*philosophie* gegeben: Die Beantwortung der Frage nach dem Warum der Sprache.

Dieser Definitionsform schließen sich die Mehrheit der Autoren von sprachphilosophischen Werken an. Sprachphilosophie betrachtet das Wesen und den Grund der Sprache sowie ihre Funktionen [Ber99, S. 15]. Um das zu verstehen, muss auch der gesellschaftliche

Kontext einer Sprache betrachtet werden, d.h. ”die Beziehungen zwischen einer Sprache und der gesamten übrigen Kultur der Gesellschaft, die diese Sprache spricht“ ist Gegenstand des Erkenntnisprozesses [Who63, S. 140, Nachwort des Übersetzers]. Auch nach Coseriu muss die Gesamtheit der Tätigkeiten von Menschen betrachtet werden, um die sprachphilosophischen Fragen beantworten zu können [Cos03, S. 13].

Unter der Vielzahl an sprachphilosophischen Gebieten sind vor allem zwei Strömungen von Bedeutung. Mit dem Aufkommen von erkenntnistheoretischen Untersuchungen und der empirischen Forschung teilte sich die Sprachphilosophie in zwei Strömungen auf [Nie96, S. 123]. Die *Philosophie der idealen Sprache* betrachtet die Konstruktion von künstlichen Sprachen. Mit dieser Sichtweise betrachten Philosophen Sprache als System und Generierung [Nie96, S. 90]. Als zweite Strömung entwickelte sich die *Philosophie der normalen Sprache* aus den Sprachspielen von Ludwig Wittgenstein heraus [Ber99, S. 149]. Diese Philosophie untersucht vor allem die Verwendung von natürlichen Sprachen [Kat69, S. 22.f].

2.3 Die Gebiete der Sprachphilosophie

Ganz im Sinne einer Philosophie, dem Streben nach dem Wesen und der Beantwortung der Frage nach dem Warum, lassen sich vor allem frühzeitliche Fragen entdecken. Bereits in der Antike betrachtete Platon in seinem Dialog ”Kratylos“ die Frage, ob die Lautgestalt der Wörter einen direkten Bezug zu ihrer Bedeutung haben, d.h. ob die Beziehung zu einer Sache natürlichen oder künstlichem Ursprungs sind [SR89, S. 297]. Die Referenzierung sprachlicher Ausdrücke wird auch in [Nie96, S. 9], und [Lyc00, S. 28]thematisiert. Auch Fragen über den Zusammenhang der Sprache zur physischen Welt und den Ursprung der Sprache lassen sich bis in die Antike zurückverfolgen [Nie96, S. 9]. Zur Philosophie der idealen Sprache gehören auch Untersuchungen über Universalsprachen [Cos03, S. 178ff.].

Durch zivilisatorischen Fortschritt im Allgemeinen und mit Entstehung der modernen Mathematik und des Computers im speziellen hat ein Wandel der Fragestellungen stattgefunden. Innerhalb der Philosophie der idealen Sprache erarbeiteten Theorien, beispielsweise die Sprachbeschreibungsverfahren von Quine und Wittgenstein [Nie96, S. 123], sind durch die neue Technik als Computermodell darstellbar und dadurch mathematisch überprüfbar geworden. Weitere Beispiele sind die Übernahme des Kompositionalitätsprinzip nach Frege in die sogenannten Montague-Grammatiken [CEE$^+$04, S. 278ff.] und die Benutzung von Grammatikteheorien nach [Cho92, S. 52] und [Kat69, S. 110], um einen Text durch einen Part-Of-Speech-Tagger mit Zusatzinformationen über dessen grammatische Struktur auszuzeichnen. Dabei kommen Markov-Modelle oder Entscheidungsbäume als Techniken zur statistischen Erkennung der grammatischen Struktur zum Einsatz [MS99, S. 345ff.]. Das Ergebnis kann mit XML-Dokumenten dargestellt werden [Kun07, S. 62]. Die moderne Linguistik profitierte enorm von sprachphilosophischen Vordenkern.

Durch diesen Fortschritt sind ehemals sprachphilosophische Überlegungen nun zu einem festen Bestandteil der Linguistik geworden. Durch weitere Fortschritte in der Psychologie sowie den Neuro- und Kognitionswissenschaften können eines Tages vielleicht auch

die letzten sprachphilosophischen Fragestellungen mit naturwissenschaftlichen Methoden beantwortet werden[2]. Die weiterhin genuinen sprachphilosophischen Gebiete beschäftigen sich demnach vor allem mit den bereits zu Beginn identifizierten Fragen nach der Herkunft von Sprache und der Referenzierung sprachlicher Ausdrücke.

2.4 Zusammenfassung

Sprache als mächtiges Instrument zur Kommunikation ist Gegenstand einer Vielzahl philosophischer und sprachwissenschaftlicher Untersuchungen. Im Sinne einer Philosophie der Sprache werden vor allem Fragen nach dem Ursprung, der Darstellung und der Funktion der Sprache gestellt. Die erblühende Mathematik gab Forschern ein breites Formalisierungsspektrum an die Hand, mit der Sprachen beschrieben werden können. Durch die technische Revolution des Computers sind viele vormals sprachphilosophische Theorien durch den Computer überprüfbar geworden. Sie haben sich dadurch zu Theorien und Methoden der Sprachwissenschaft und speziell der Computerlinguistik geworden.

Als Fazit wird geschlossen, dass die modernen Sprachphilosophischen Betrachtungen nur begrenzt eingesetzt werden können, um zum Ziel dieses Artikels, der Erarbeitung der Sprachdefinitionsbestandteile beitragen zu können. Stattdessen sind die Theorien der Linguistik vorzugsweise zu studieren.

3 Herleitung der Sprachdefinitionsbestandteile für die Linguistik und die Informatik

Sprachphilosophen, Linguistiker und Computerlinguistiker beschreiben übereinstimmend die drei Bestandteile von Sprachen: Syntax, Semantik und Pragmatik [Mar96, S. 4], [Kut75, S. 17ff.], [SR89, S. 299], [GSCK04, S. 280f.], [Lou94, S. 20] [CEE$^+$04, S. 149f.]. In einer Kurzzusammenfassung bezeichnet die Syntax die Struktur der Sprache, die Semantik die Bedeutung von sprachlichen Äußerungen und die Pragmatik die über die Äußerungen hinausgehenden Bedeutungsaspekte. Wie zu Begin am Beispiel der DSL beschrieben, ist jedoch die genaue Definition dieser Begriffe offen. In den folgenden Abschnitten werden diese Begriffe jeweils für die Linguistik, die Informatik (Programmiersprachen) und den DSL erarbeitet[3]. Eine zusammenfassende Diskussion erläutert abschließend die Gemeinsamkeiten und die Unterschiede dieser Definitionen.

[2] So ist zum Beispiel das FOXP2 Gen von großer Bedeutung für die sprachlichen Fähigkeiten eines Menschen [MBS$^+$05].

[3] Ein Hinweis zu Terminologie. Ein Begriff kann in mehrere Unterbegriffe aufgeteilt sein. Die Unterbegriffe formen Dimensionen des Oberbegriffs. Dimensionen bezeichnen entweder verschiedene Blickwinkel auf den zentralen Begriff oder können im Sinne einer Systematik entweder mono- oder polyhierarchische Klassen aufspannen. Unterbegriffe können rekursiv wieder aus anderen Unterbegriffen bestehen. Eine Differnzierung steht für die Unterteilung eines Begriffs in Dimensionen

3.1 Definition in der Linguistik

Die moderne Linguistik benutzt den Computer und eine Vielzahl von Textsammlungen, um eine genaue Aussage hinsichtlich der Struktur und der Bedeutung ihrer Inhalte geben zu können. Die Verfahren, die dabei zum Einsatz kommen, benötigen eine wohlstrukturierte Terminologie, um die Sprachbestandteile genau zu beschreiben. Im Folgenden ist das Ergebnis der Begriffsanalyse für die Linguistik zu finden.

Syntax: Bei der Beschreibung der Syntax einer Sprache wird zuerst von einem Alphabet zulässiger *Symbole* ausgegangen. Aus den Symbolen heraus werden die *Wörter* gebildet. Wörter bestehen aus *Silben*, welche die Lautgestalt darstellen, und *Morpheme*, den kleinsten Bedeutungsträgern. Silben und Morpheme müssen können identisch sein, sind es aber meistens nicht [KRM06, S. 37]. Die Menge aller Wörter beschreibt den "Wortschatz der Sprache" [Kat69, S. 115].

Diese Wortmenge muss nun in einen sinnvollen Zusammenhang gebracht werden. Damit Sprache für den Austausch von Informationen verwendet werden kann, müssen Regeln für eine Strukturierung aufgestellt werden. Diese Strukturierung beschränkt sich auf die wichtigsten linguistischen Begriffe, eine vollständige Beschreibung der Grammatik natürlicher Sprachen ist z.B. im Duden der deutschen Sprache[4] zu finden. Jedes Wort gehört zu einer *Wortart* wie den Substantiven, Verben, Adjektiven und Artikel [Kut75, S. 208ff.]. In der Zusammensetzung bilden Wörter einzelne *Phrasen*. Auch Phrasen haben einen Typ, zum Beispiel die Nominal- oder Adverbialphrasen [GHS93, S. 169]. Die erlaubten Kombinationen von Phrasen werden in *Sprachschemata* beschrieben. Damit ein Satz eine gültige Aussage ist, muss seine Form also einem Sprachschemata der benutzten Sprache entsprechen.

Semantik: Die genaue Definition der Semantik ist bei den Philosophen ein großer Streitpunkt, der in seiner Vielschichtigkeit hier nicht wiedergegeben werden kann. Ein gemeinsamer Konsens kann über den Begriff der Aussage gefunden werden. Sowohl Kutschera als auch Lycan sehen in der Semantik die Bedeutung einer Aussage [Kut75, S. 18], und Lycan führt zusätzlich die Beziehung zwischen einer Aussage und der umgebenden Welt einer Äußerung auf [Lyc00, S. 165]. Das deutet bereits auf mehrere Stufen der Definition von Semantik hin. Die dreistufige Semantik von [CEE$^+$04, S. 276] bildet dabei die begriffliche Grundlage für die folgende Aufteilung.

Wie bereits festgestellt besteht eine Aussage aus einzelnen Wörtern. Jedes Wort hat für sich genommen eine Bedeutung, es ist der Zeiger für ein bestimmtes Objekt oder referenziert eine Handlung. Das wird als *lexikalische Semantik* bezeichnet. Die nächste Stufe betrachtet dann den gesamten Satz. Durch die Kombination der Wörter verschmelzen die Wortsemantiken zu einer Ganzheit, die *Satzsemantik* genannt wird. Damit jedoch auch die Aussage vollständig verstanden wird, muss der Kon-

[4] Im Duden werden noch feinere Unterschiede vorgenommen. Wortarten werden in flektierbare und nicht flektierbare unterschieden. Zudem hat jedes Wort mehrere grammatische Merkmale, wie Person, Numerus, Genus usw. [KRM06, S. 146ff.]

text, definiert durch den umgebenden Text, betrachtet werden. Dies geschieht durch die *Diskurssemantik*.

Pragmatik: Die Pragmatik beschreibt die sprachliche Situation, in der Aussagen getroffen werden. Kutschera spricht von pragmatischen Umständen, die die sprachlichen Konventionen betreffen, etwa die Definition von Sprecher und Adressat [Kut75, S. 19]. Diese entspricht einer linguistischen Definition, die auch Lycan vertritt [Lyc00, S. 165]. Pragmatik definiert den grundsätzlichen „Handlungscharakter“ einer Aussage [SR89, S. 299].

Die sprachliche Situation ist somit Teil einer Aussage und kann auch Einfluss auf die Bedeutung haben, wie es die Diskurssemantik definiert. Die Unterscheidung von Semantik und Pragmatik wird durch folgende Überlegungen getroffen: Nachdem die allgemeine Bedeutung einer (isolierten) Aussage durch die Semantik bestimmt wurde, definiert die Pragmatik sämtliche darüber hinaus gehende Bedeutungsaspekte [CEE$^+$04, S. 333]. Weitere Unterklassen des Pragmatikbegriffs ließen sich in der Literatur nicht finden.

Die Syntax ist das am besten ausgebaute Sprachbestandteil, welches die Linguistik erarbeitet hat. Durch die begriffliche Präzisierung kann der strukturelle Aufbau von Sprachen sehr genau vorgenommen werden. Gerade für die deutsche Grammatik ist eine sehr differenzierte Terminologie erarbeitet wurden. Dahingegen ist die Entwicklung der Semantik und speziell der Pragmatik noch nicht weit fortgeschritten. Statt die Strukturen dieser Begriffe zu beschreiben, ist in der Literatur eine Vielzahl an Sprachbeschreibungsverfahren entwickelt wurden[5]. Eine Betrachtung dieser ist jedoch nicht die Aufgabe dieses Artikels.

In der Abbildung 1 werden die Begriffszusammenhänge untereinander noch einmal dargestellt.

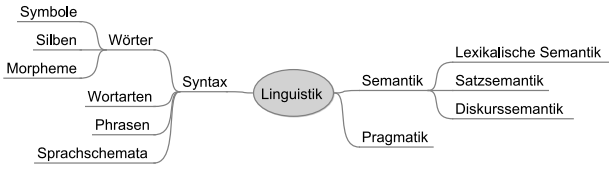

Abbildung 1: Sprachdefinitionsbestandteile in der Linguistik

3.2 Definition in der Informatik

Die Informatik als noch junge Wissenschaftsdisziplin hat bereits eine beachtliche Vielfalt an Forschungsgebieten entwickelt. Die Basis bildet die theoretische Informatik und

[5] In [GHS93, S. 317ff.] wird die Semantik mit so unterschiedlichen Methoden wie die Aussagenlogik, Kompositionalität, Prädikation, Quantoren und Bedeutungspostulate beschrieben. [CEE$^+$04, S. 339ff.], ein Werk über Computerlinguistik, führt die Methoden der Anaphernresolution, Präsuptionsverarbeitung, Fokus und Benutzermodellierung zur Erklärung der Pragmatik auf

die Mathematik, auf deren Fundament sich Künstliche Intelligenz, Software Engineering, Software Technik, Technische Informatik und differenzierte Arten von Angewandter Informatik stützen. Innerhalb der Software Technik wird die Entwicklung von Programmiersprachen untersucht. Im Folgenden wird überwiegend Literatur aus der Software Technik verwendet, um die Sprachbestandteile von Programmiersprachen zu definieren.

Syntax: In der Informatik werden die Syntaxdefinitionsbestandteile in zwei Klassen aufgeteilt, um die vielfältigen syntaktischen Phänomene beschreiben zu können: Die *Microsyntax* und die *Macrosyntax* [Wat91, S. 34f.]. In der Microsyntax werden die kleinsten Bedeutungsträger einer Programmiersprache, die *Token* beschrieben. Token beschreiben die zulässigen Zeichenketten innerhalb einer Sprache [Lou94, S. 114]. Gleichsam wie in der Linguistik werden die Token einem speziellen *Tokentyp* zugeordnet. Tokentypen sind Literale, Identifier, Operatoren und weitere [Wat91, S. 35] [PZ, S. 74ff.]. Die erlaubten Token bilden ähnlich wie in der Linguistik den Wortschatz der Sprache. In der Macrosyntax werden die möglichen Sprachschemata, in der Informatik *Statements* genannt, beschrieben. Ein Statement legt die Struktur gültiger Aussagen innerhalb eine Sprache fest [PZ, S. 74ff.], [FWH01, S. 120f.]. Statements können dabei *Expressions*, *Declarations* und *Commands* sein [Wat91, S. 35]. Zur Unterscheidung dieser Begriffe: Statements beschreiben allgemeine Anweisungen an den Computer, die zu einer System-Zustandsänderung führen können. Expression bezeichnen Anweisungen mit einem Rückgabewert und Commands sind kleine „Bausteine“ der Statements [WC01, S. 78].

Diese Syntaxdefinitionsbestandteile können in zwei verschiedenen Realisierungsformen auftreten. Als *Concrete Syntax* wird die Darstellung von Anweisungen in einer Form beschrieben, wie sie ein Programmierer eingibt. Durch den Kompilierungsprozess wird die Syntax in eine für Maschinen besser zu bearbeitendes Format gebracht, die *Abstract Syntax*. Sie wird beispielsweise durch Syntax Trees realisiert [Pie02, S. 53].

Semantik: Der Semantik-Begriff in der Informatik ist dem der Linguistik ähnlich, wie Louden feststellt: Semantik beschreibt, was Sprachkonstrukte tun [Lou94, S. 125]. Eine genauere Begriffsanalyse brachte nur einie einzige Differenzierung: Die Unterteilung in statische und dynamische Semantik [Mey91, S. 51] [Seb99, S. 126]. Demnach beschreibt die statische Semantik zusätzliche syntaktische Eigenschaften, die sich durch die damals gebräuchlichen Mittel wie der BNF nicht darstellen ließen. Sebesta schreibt explizit von ”legal form of programms“ als die Aufgabe der statischen Semantik [Seb99, S. 126]. Da es nichts mit einer Semantik zu tun hat, ist diese Differenzierung nicht zulässig. Im Gegensatz beschreibt die dynamische Semantik die Bedeutung einzelner Statements und Expressions und ist damit die eigentliche Semantik.

Über diese Unterscheidung hinaus lässt sich in der Literatur keine weitere Begriffsdifferenzierung feststellen. Die fehlende terminologische Basis wird auch von anderen Autoren als Problem dargestellt [WC01, S. 297] [PZ, S. 125f.] [Seb99, S. 132]. Die fehlende Basis kann als Ursache für eine große Vielfalt an Sprachbeschreibungsverfahren angesehen werden. Da die Aufgabe dieses Berichts eine reine

Begriffsanalyse ist, mit dem Ziel, die verfahrensunabhängige Terminologie zu entwickeln, wird an dieser Stelle lediglich ein Überblick zu den Verfahren gegeben. Die Erarbeitung des Begriffs Semantik durch die Analyse der Verfahren ist zukünftigen Forschungen vorbehalten.

Nach Louden werden drei Möglichkeiten unterschieden. Das *Referenzhandbuch der Sprache* listet die Verwendung und Bedeutung der Syntaxelemente auf. Mit einem *Compiler* kann die Wirkungsweise eines Programmes getestet werden. Und drittens können *formale Definition* zum Einsatz kommen [Lou94, S. 125f.]. Die Methoden unterscheiden sich durch den Grad der Genauigkeit, mit der Wirkungsweise beschreiben wird und wie die Semantik interpretiert wird. Für formale Definitionen werden drei Arten unterschieden: Die denotationale und axiomatische Form [WC01, S. 297] [Seb99, S. 134ff.], sowie zusätzlich die operationale Semantik [Bru02, S. 14], [Seb99, S. 132]. Die *Denotationale Semantik* beschreibt die Auswirkungen, die die Ausführung einer Programminstruktion hat. Dazu werden mathematische Funktionen und ihre Ergebnisse zur Modellierung verwendet [WC01, S. 298]. Bei der *Axiomatischen Semantik* werden logische Symbole und deren Regeln benutzt. Mit Hilfe von Pre- und Post-Konditionen, bezeichnet als Assertions [Seb99, S. 135], wird die Auswirkung von Programmausdrücken mathematisch beschrieben [WC01, S. 301]. Die *Operationale Semantik* gibt dagegen eine Menge an Instruktionen für einen Compiler an, mit dem das Programm ausgeführt wird [Bru02, S. 14]. Beim Kompilieren werden die Ausdrücke der Programmiersprache in eine Zwischensprache und schließlich in ein sogenanntes Instruction Set des Computer, z. B. MIPS, übertragen [PJLH07, S. 48.f]. Diese Instruktionen erbringen einfache Rechenoperationen auf dem Computer.

Pragmatik: Der zentrale Begriff der Pragmatik spielt in Informatikbüchern eine unbedeutende Rolle, er wird nicht als Bestandteil einer Sprachdefinition gesehen. Pragmatik kann zur Beschreibung von Aspekten des Software-Engineerings, die sich mit der Entwicklung und Anwendung der Programme befassen, benutzt werden [Ort05, S. 190f.]. Demnach wird der Handlungscharakter von Sprachartefakten der Programmiersprachen für Entwickler und Nutzer separat betrachtet. Die *Entwicklerpragmatik* stellt die funktionsfähige Anwendung dar, die etwa einen Teil der Realität als Modell im Computer abbildet. Hingegen steht die *Anwendungspragmatik* für den Zweck, den die Benutzer mit der Anwendung erreichen: Etwa betriebswirtschaftliche Buchungsvorgängen vorzunehmen. In beiden Fällen wird die Pragmatik nur über das vollständige Programm beschreibbar.

Die Informatik hat zur Erklärung der Struktur von Programmiersprachen ebenfalls eine gute Differenzierung ausgearbeitet. Bei der Semantik allerdings wurde festgestellt, dass keine einheitliche Terminologie in der Literatur entwickelt ist. Verschiedene Verfahren versuchen zu erklären, was die Bedeutung von Ausdrücken einer Programmiersprache ist. Die Pragmatik spielt bei reinen Sprachbeschreibungen eine untergeordnete Rolle. Das ist erklärbar, wenn man die Aufgabe einer solchen Beschreibung betrachtet: Das Formulieren von Regeln, wie die Sprache benutzt wird und was die Bedeutung einzelner Ausdrücke sind. Pragmatik kann daher nur beschreiben, was die Bedeutung eines durch Regeln der

Syntax gebautes und nach den Regeln der Semantik interpretiertes Programm für den Entwickler und den Benutzer darstellt.

Die folgende Abbildung 2 zeigt die begrifflichen Zusammenhänge als Zusammenfassung.

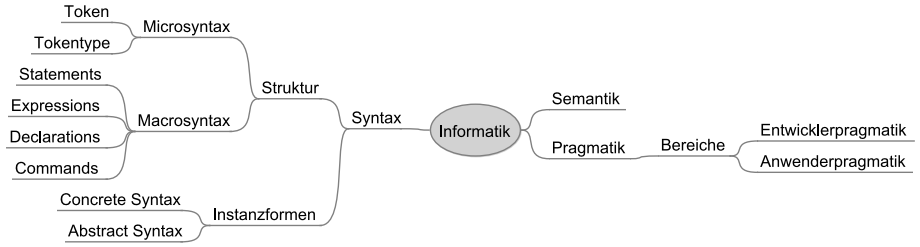

Abbildung 2: Sprachdefinitionsbestandteile in der Informatik

3.3 Diskussion der Sprachbestandteile

In einer abschließenden Betrachtung sind Parallelen zwischen der Art und Weise, wie die Sprachbestandteile in der Linguistik und in der Informatik definiert wurden, zu ziehen. In beiden Gebieten stellt die Syntax den am weitesten entwickelten Bestandteil einer Sprachbeschreibung dar. Die präzise Erklärung des Aufbaus von Sätzen in natürlichen Sprachen und von Statements in Programmiersprachen ist notwendig, um in Richtig und Falsch unterscheiden zu können. Dabei ist die Vielfalt bei den natürlichen Sprachen ungemein höher, da ein Wort durch bis zu sieben verschiedene grammatische Merkmale beschrieben werden kann [KRM06, S. 131]. Dies ist begründet in der großen Vielfalt von Äußerungen und der damit einhergehenden Ausdrucksmächtigkeit einer Sprache. Wenn die Bedeutung analysiert wird, so verblassen beide im Gegensatz zu der Syntax. Die Semantik in der Linguistik kennt eine dreistufige Unterteilung. Wenn auch hier die Methoden analysiert werden, kann geschlussfolgert werden, dass diese Differenzierung erst durch die Entwicklung der Sprachbeschreibungsverfahren und deren Einigung auf eine Begriffsbasis stattgefunden hat. Eine solche Einigung hat in der Informatik noch nicht stattgefunden; die einzelnen Verfahren benutzen eigenständige Terminologien, die noch nicht kompatibel zueinander sind. Die Unterteilung aus der Linguistik kann auf die Informatik übertragen werden, indem die Semantik für Token, für Statements und schließlich für größere Programmblöcke mit geeigneten Mitteln beschrieben wird. Doch welche Mittel das sind und wie sie eingesetzt werden können, ist außerhalb dieses Beitrages. Daher muss zu diesem Zeitpunkt der Schluss gezogen werden, dass die genaue Definition der Semantik einer Programmiersprache sich erst aus der Analyse und Systematisierung verschiedener Beschreibungsmöglichkeiten ableiten lässt. Dies ist zukünftigen Forschungen vorbehalten. Auch für die Pragmatik der Linguistik gilt die Schlussfolgerung, dass eine Vielzahl an Methoden mit unsicherer Terminologie das Begriffsfundament verwehrt. Die beiden abgegrenzten Pragmatikdimensionen der Informatik spielen bei einer reinen Sprachbeschrei-

bung keine wichtige Rolle.

4 Definition der Sprachbestandteile für Domänenspezifische Sprachen

Bereits in der Einleitung wurde das große Potential von DSL festgestellt, die Probleme des Software Engineering zu lösen. Sie können vielfältig zur Verbesserung und zunehmenden Automatisierung der Software-Erstellung beitragen, kapseln das Domänenwissen in einer sprachlichen und wiederverwendbaren Form und sind durch den Bezug auf eine Domäne sehr kompakt. Vielfältige Definitionsaspekte sind in der Literatur zu finden [Nei80, S. 6] [Ben86, S. 711] [Hud98, S. 134] [vDKV00, S. 26] [Hee00, S. 2] [Beu04, S. 40ff.]. Folgende Definition fasst diese Aspekte zusammen:

> Eine Domänenspezifische Sprache ist eine künstliche, vom Computer interpretier- und ausführbare Sprache, welche die Konzepte einer Anwendungsdomäne sprachlich fixiert. Sie beruht auf einer Fachsprache und kann deren syntaktische Form benutzen.

Ein Fachsprache (Beschreibung der Domänenkonzepte) und eine Programmiersprache (vom Computer interpretier- und ausführbar) verschmelzen in der DSL ineinander. Diese besondere Dualität wird bei der Erarbeitung der Sprachbestandteile berücksichtigt. Sie erlaubt es, Differenzierungen der Gebiete der Linguistik und der Informatik zu integrieren.

Syntax: Im Kern bleibt die Syntaxdefinition einer DSL den vorgestellten Kriterien der Informatik gerecht. Die Betrachtung des Charakters einer Fachsprache aber führt zu zwei Besonderheiten. Eine Fachsprache orientiert sich zumeist an natürlichen Sprachen. Dies ermöglicht einen intuitiven Zugang zu der Sprache und adressiert psychologische Motive der Sprecher. Davon kann die DSL profitieren - sie übernimmt syntaktische Strukturen der Fachsprache und kann somit ohne viel Aufwand von den Sprechern der Fachsprache beherrscht werden. Dies wird als *Fachsprachensyntax* bezeichnet. Sollte eine DSL dagegen auf einer gewöhnlichen Programmiersprache beruhen, so kann für die Einbindung der Konzepte eine zweite Besonderheit festgestellt werden. Die Konzepte stehen in verschiedenen Beziehungen zueinander, die sprachlich fixiert sind. Variationen in der Syntax (etwa grammatische Eigenschaften) lassen direkt auf Änderungen in der Semantik schließen. Wenn der Numerus geändert wird, so werden mehrere Instanzen der Konzepte angelegt. Oder wenn das Tempus geändert wird, dann beschreiben Aussagen der DSL Zustände der Vergangenheit oder Wünsche für die Zukunft. Diese Besonderheit wird durch die *Konzeptsyntax* abgebildet. Die letztendlich resultierende Syntaxbeschreibung einer DSL muss noch um die notwendigen programmiersprachlichen Konstrukte ergänzt werden. Die vollständige Syntaxbeschreibung wird als *Sprachsyntax* bezeichnet.

Semantik: Kern einer DSL sind die der Fachsprache entnommenen Begriffe (auch als Konzepte im philosophischen Sinn bezeichnet). Konzepte beschreiben konkrete oder

abstrakte Entitäten innerhalb einer Anwendungsdomäne. Sie stehen in vielfältigen Klassifikations- und Bedeutungsbeziehungen zueinander. Die Konzepte als solche besitzen eine Semantik, die als *Konzeptsemantik* bezeichnet wird. Andere Konstrukte der DSL, die den von der Informatik identifizierten Wortarten wie Prozeduren, Funktionen und Variablen entsprechen, haben ebenfalls eine Semantik. Es muss unterschieden werden, ob diese Worte Teil der Fachsprache und damit durch die Konzeptsemantik definiert sind, oder ob sie Teile der als Basis verwendeten Programmiersprache sind und damit eine *Programmsemantik* besitzen. Eine zusammenhängende Aussage innerhalb einer DSL besteht demnach aus Worten der Fachsprache und einer Programmiersprache. Im Sinne der Diskurssemantik aus der Linguistik wird die Fusion der Einzelsemantiken als *Sprachsemantik* bezeichnet. Sprachsemantik ist eine symbolische Repräsentation der Aussage, die vom Computer interpretiert werden kann, und entspricht damit der Semantik von Programmiersprachen. Diese aber muss, wie zuvor gezeigt, noch erarbeitet werden.

Pragmatik: Wie beschrieben spielt die Betrachtung der Pragmatik innerhalb der Informatik keine wichtige Rolle, daher stimmt die Definition der Pragmatik von DSL und der Informatik überein. Auch sie unterteilt den Handlungscharakter von Spracharte-fakten für Entwickler und Nutzer.

Bedingt durch die Fachsprache hat die DSL bei Syntax und Semantik weitere Dimensionen bekommen, die zur präzisen Beschreibung dieser Sprachbestandteile benutzt werden können. Im Wesentlichen aber werden die Begriffsdifferenzierungen der Informatik übernommen. Die folgende Abbildung 3 zeigt die erarbeitet Definition der Sprachbestandteile für DSL.

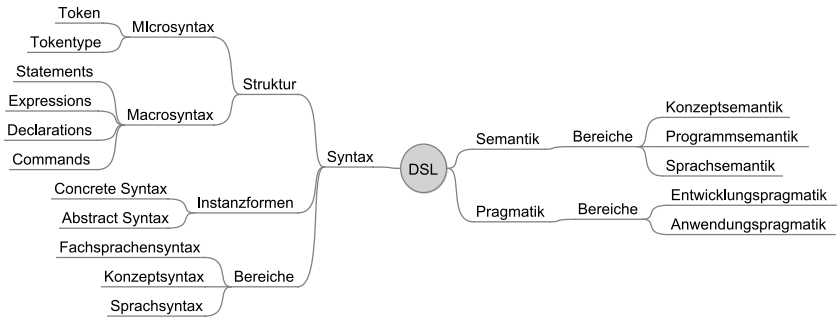

Abbildung 3: Sprachdefinitionsbestandteile von Domänenspezifischen Sprachen

5 Zusammenfassung & Ausblick

Dieser Forschungsbericht hat gezeigt, wie die zentralen Begriffe von Syntax, Semantik und Pragmatik für Domänenspezifische Sprachen definiert werden, ausgehend von den

sprachphilosophischen Wurzeln der Informatik. Dazu wurde gezeigt, was Sprachphilosophie ist und wie sie sich von linguistischer Forschung unterscheidet. Es wurde festgestellt, dass aufgrund des technologischen Fortschritts ehemals sprachphilosophische Fragestellungen nun von der (Computer)Linguistik behandelt werden. Darauf wurde die Definition der zentralen Begriffe aus der Perspektive der Linguistik, der Informatik (speziell Programmiersprachen) und der Domänenspezifischen Sprachen erarbeitet. Abbildungen fassten die Definitionen zusammen.

Die dargestellten Erkenntnisse bilden lediglich die Grundlage für weitere Forschungen. Wie gezeigt, stellt vor allem die Erarbeitung einer Semantikdefinition in der Informatik ein wichtiges Gebiet dar. Erst wenn die Vielzahl der gezeigten Semantikbeschreibungen in eine Systematik gebracht werden kann, ist eine Definition des Semantikbegriffs für Programmiersprachen und für DSL möglich. Offen bleibt auch die Frage, wie die speziellen Syntax- und Semantik-Dimensionen von DSL konkret ausgestaltet und bei der Konstruktion einer solchen Sprache benutzt werden. Eine weitere Möglichkeit stellt die Übertragung sowohl der Semantikdefinition als auch der Semantikbeschreibung von der Linguistik auf die Informatik dar.

Weitere Forschungsgebiete wurden durch die geschilderten Problemfelder identifiziert. Demnach wird eine Taxonomie zur Klassifikation von DSL Arten erstellt und ein Referenzprozess zur Konstruktion von Domänenspezifischen Fragen erarbeitet. In der Informatik ist es zudem interessant, ob mit der formalen Beschreibung der Sprachdefinitionsbestandteile eine Analyse hinsichtlich der Ausdrucksmöglichkeiten von Sprachen und deren mögliche Äquivalenz dargestellt werden kann. Solche Überlegungen können zu einem qualitativen Vergleich verschiedenen (Modellierungs)Sprachen verwendet werden. All diese Erkenntnisziele sind zukünftigen Forschungen vorbehalten.

Literatur

[BCDVM02] A. Bryant, A. Catton, K. De Volder und G.C. Murphy. Explicit programming. In *Proceedings of the 1st international conference on Aspect-oriented software development*, Seiten 10–18. ACM Press New York, NY, USA, 2002.

[Ben86] Jon Louis Bentley. Little Languages. In *Communications of the ACM*, Jgg. 29, Seiten 711–721, 1986.

[Ber99] C. Bermes, Hrsg. *Sprachphilosophie*. Verlag Carl Alber, Freiburg, München, 1999.

[Beu04] D. Beuche. *Composition and Construction of Embedded Software Families*. Dissertation, Otto-von-Guericke-Universität Magdeburg, 2004.

[BJNR06] M. Broy, M. Jarke, M. Nagl und H. D. Rombach. Strategische Bedeutung des Software Engineering in Deutschland. In *Informatik Spektrum*, Jgg. 29, Seiten 210–221. 2006.

[Bor95] A. Borgida. Description logics in data management. In *IEEE Transactions on Knowledge and Data Engineering*, Jgg. 7, Seiten 671–682. 1995.

[Bru02] K. B. Bruce. *Foundations of Object-Oriented Languages: Types and Semantics*. MIT Press, Cambridge, London, 2002.

[CE00] K. Czarnecki und U.W. Eisenecker. *Generative programming: methods, tools, and applications*. ACM Press/Addison-Wesley Publishing Co. New York, USA, 2000.

[CEE+04] K.-U. Carstensen, C. Ebert, C. Endriss, S. Jekat, R. Klabunde und H. Langer, Hrsg. *Computerlinguistik und Sprachtechnologie. Eine Einführung*. Spektrum Akademischer Verlag, München, 2. Auflage, 2004.

[Cho92] N. Chomsky. *Sprache und Geist*. Suhrkamp Taschenbuch Verlag, Frankfurt am Main, 5. Auflage, 1992.

[Cos03] E. Coseriu. *Geschichte der Sprachphilosophie von der Antike bis zur Gegenwart 1. Von den Anfüngen bis Rousseau.* A. Francke Verlag, Tübingen, Basel, 2003.

[Dre06] A. Dreiling. On the impact of the 'linguistic turn' on research in information systems. In J. Ljunberg und M. Andersson, Hrsg., *Proceedings of the Fourteenth European Conference on Information Systems*, Seiten 530–545. 2006.

[FN08] K. Fielenbach und B. Niehaves. Theories of Language in IS Reseach - A Review. In J. Becker, H. Krcmar und B. Niehaves, Hrsg., *Arbeitsberichte des Instituts für Wirtschaftsinformatik*, Jgg. 120, Seiten 87–110, 2008.

[FWH01] D. P. Friedman, M. Wand und C. T. Haynes. *Essentials of Programming Languages*. The MIT Press, Cambridge, London, 2. Auflage, 2001.

[GHS93] G. Grewendorf, F. Hamm und W. Sternefeld. *Sprachliches Wissen: Eine Einführung in moderne Theorien der grmmatischen Beschreibung*. Suhrkamp Verlag, 6. Auflage, 1993.

[GSCK04] J. Greenfield, K. Short, S. Cook und S. Kent. *Software Factories: Assembling Applications with Patterns, Models, Frameworks and Tools*. Wiley Publishing, Inc., 2004.

[Hee00] J. Heering. Application Software, Domain-Specific Languages, and Language Design Assistants. In *The Computing Research Repository*, Seiten 1–6, 2000.

[Hud98] Paul Hudak. Modular Domain Specific Languages and Tools. In P. Devanbu und J. Poulin, Hrsg., *Proceedings: Fifth International Conference on Software Reuse*, Seiten 134–142. IEEE Computer Society Press, 1998.

[Kat69] J. J. Katz. *Philosophie der Sprache*. Suhrkamp Verlag, Frankfurt, 1. - 3. tsd. Auflage, 1969.

[KLM+97] G. Kiczales, J. Lamping, A. Menhdhekar, C. Maeda, C. Lopes, J.-M. Loingtier und J. Irwin. *Aspect-Oriented Programming*, Jgg. 1241, Seiten 220–242. Springer-Verlag, Berlin, Heidelberg, and New York, 1997.

[KRM06] K. Kunkel-Razum und F. Münzberg, Hrsg. *Duden: Die Grammatik*. Bibliographisches Institut & F.A. Brockhaus AG, Mannheim ; Leipzig ; Wien ; Zürich, 7. Auflage, 2006.

[Kun07] M. Kunze. *Linguistische Analysen für die semantische Auszeichnung natürlichsprachlicher Dokumente*. Dissertation, Otto-von-Guericke-Universität Magdeburg, München, 2007.

[Kut75] F. von Kutschera. *Sprachphilosophie*. Wilhelm Fink Verlag München, München, 2. Auflage, 1975.

[Lou94] K.C. Louden. *Programmiersprachen: Grundlagen, Konzepte, Entwurf*. International Thomson Publishing GmbH, 1994.

[Lyc00] W. G. Lycan. *Philosophy of Language: A Contemporary Introduction*. Routledge, New York, 2000.

[Mar96] A. P. Martinich. *The Philosophy of Language*. Oxford University Press Inc, USA, 3. Auflage, 1996.

[MBS+05] K. D. MacDermot, E. Bonora, N. Sykes, A.-M Coupe, C. S. L. Lai, S. V. Vernes, F. Vargha-Khadem, F. McKenzie, R. L. Smith, A. P. Monaco und S. E. Fisher. Identification of FOXP2 Truncation as a Novel Cause of Developmental Speech and Language Deficits. In *American Journal of Human Genetics*, Jgg. 76, Seiten 1074–1080. 2005.

[Mey91] B. Meyer. *Introduction to the Theory of Programming Languages*. Prentice Hall, 1991.

[MS99] C. D. Manning und H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, London, 1999.

[Nei80] J.M. Neighbors. *Software Construction Using Components*. Dissertation, University of California, Irvine, 1980.

[Nie96] E. Niebel. *Philosophia Propaedeutica, Was ist Sprache?* Verlag Moritz Diesterweg, Frankfurt am Main, 1996.

[Ort02] Erich Ortner. Sprachingenieurwesen Empfehlung zur inhaltlichen Weiterentwicklung der (Wirtschafts-)Informatik. In *Informatik Spektrum*, Jgg. 25, Seiten 39–51. 2002.

[Ort05] E. Ortner. *Sprachbasierte Informatik . Wie man mit Würtern die Cyber-Welt bewegt*. Edition am Gutenbergplatz, Leipzig, 2005.

[Pie02] B. C. Pierce. *Types and Programming Languages*. MIT Press, Cambridge, London, 2002.

[PJLH07] D. Patterson und J. L. John L. Hennessy. *Computer Organization and Design: The Hardware/Software Interface*. Morgan Kaufmann, 3. Auflage, 2007.

[Pre97] C. Prehofer. Feature-Oriented Programming: A Fresh Look at Objects. In Aksit M. und Matsuoka S., Hrsg., *Lecture Notes in Computer Science. ECOOP'97 - Object-Oriented Programming*, Jgg. 1241, Seiten 419–443. 1997.

[PT07] F. Padberg und W. Tichy. Schlanke Produktionsweisen der modernen Softwareentwicklung. In H. U. Buhl und W. Künig, Hrsg., *Wirtschaftsinformatik*, Jgg. 49, Seiten 162–170. 2007.

[PZ] T. Pratt und M. Zelkowitz. *Programming Languages: Design and Implementation*. Prentice Hall, 4. Auflage.

[Seb99] R. W. Sebesta. *Concepts of Programming Languages*. Addison Wesley, Reading, Harlow et. al., 3. Auflage, 1999.

[SR89] H. Seiffert und G. Radnitzky, Hrsg. *Handlexikon zur Wissenschaftstheorie. Studienausgabe*. Ehrenwirth Verlag, München, 1989.

[Sta95] Standish Research Group. Chaos Report, 1995.

[VDBHKO02] MGJ Van Den Brand, J. Heering, P. Klint und PA Olivier. Compiling language definitions: the ASF+ SDF compiler. In *ACM Transactions on Programming Languages and Systems (TOPLAS)*, Jgg. 24, Seiten 334–368. ACM Press New York, NY, USA, 2002.

[vDKV00] A. van Deursen, P. Klint und J. Visser. Domain-Specific Languages: An Annotated Bibliography. In *ACM SIGPLAN Notices*, Jgg. 35, Seiten 26–36, 2000.

[Wat91] D. A. Watt. *Programming Language Syntax and Semantics*. Prentice Hall, Cambridge, 1991.

[WC01] L. Wilson und R. Clark. *Comparative Programming Languages*. Addison Wesley, 3. Auflage, 2001.

[WH07] T. Wilde und T. Hess. Forschungsmethoden der Wirtschaftsinformatik. In H. U. Buhl und W. Künig, Hrsg., *Wirtschaftsinformatik*, Jgg. 49, Seiten 280–287, 2007.

[Who63] B. L. Whorf. *Sprache, Denken, Wirklichkeit*. Rowohlt Taschenbuch GmbH, 1963.

[Yet08] F. Yetim. Form Communicative Action Theory to Socio-Technical Artifacts: Presentation of Three System Prototypes. In J. Becker, H. Krcmar und B. Niehaves, Hrsg., *Arbeitsberichte des Instituts für Wirtschaftsinformatik*, Jgg. 120, Seiten 27–47. 2008.