Model Extraction: Transformation von Benutzeranforderungen in UML-Modelle mit Hilfe statistisch-linguistischer Methoden und Heuristiken

André Mai

kommfinanz-Software GbR
Breitkopfstraße 4
Albert-Einstein-Str.21
04317 Leipzig
am@kommfinanz.de

Stefan Gerber

Eitel-Fritz-Straße 33
14129 Berlin

Abstract: Benutzeranforderungen werden von den Benutzern häufig natürlichsprachlich formuliert, obwohl die UML als Quasistandard in Analyse- und Design von Softwareanwendungen gilt. Durch den unvermeidlichen Medienbruch sind Missverständnisse vorprogrammiert, die durch eine Transformation von natürlichsprachlichen Texten in semiformale Modelle vermieden werden können. Im Beitrag werden existierende Möglichkeiten zur Extraktion von UML-Klassenmodellen aus Benutzeranforderungen, statistisch-linguistische Analysen und Heuristiken untersucht und zu einer neuen Methode kombiniert.

Begriffe: MDA, Transformation von Modellen, statistisch-linguistische Verfahren, Information Retrieval

1. Darstellung und Interpretation von Benutzeranforderungen

1.1 Darstellung von Benutzeranforderungen

Die Effizienz von Softwareentwicklungsprojekten und die Akzeptanz der entstehenden interaktiven Anwendungen lässt sich steigern, wenn die späteren Benutzer als Fachexperten in die Entwicklung einbezogen werden [Wiem03]. Dafür müssen u. a. Benutzeranforderungen erhoben, dokumentiert, umgesetzt und getestet werden [Part98].

Erfahrungsgemäß sind Fachexperten nicht ausreichend in den Methoden des Software-Engineerings geschult. Sie stellen ihre Anforderungen mit Hilfe natürlichsprachlicher Dokumente [Rupp01a], [Rupp01b], [Rupp03] oder mit Hilfe von Prozessmodellen dar [Sche98], während im Software-Engineering die UML Anwendung findet, u. a. [OMG00, S. 2], [OMG01, S. 6], [Jec+04a, S. 323], [Jec+04b, S. 10], [Rump05].

Das gemeinsame Verständnis der dem zu entwickelnden System zu Grunde liegenden Fachterminologie [Heye02, S. 1] hilft, die Fachexperten in das Projekt zu integrieren, und Missverständnisse bei der Interpretation von Anforderungen durch die Entwickler zu vermeiden [Ortn00a], [Ortn00b]. Sowohl die natürlichsprachliche Darstellung als auch UML-Diagramme haben ihre Berechtigung, die erste auf Grund ihrer Verständlichkeit für ungeübte Analysten, die zweite wegen ihrer höheren Präzision. Der Einsatz mehrerer Methoden führt jedoch zu Konsistenzproblemen, wenn die jeweiligen Dokumente oder Modelle per Hand abgeglichen werden müssen [RuSc04, S. 2].

Um Konsistenzprobleme zu vermeiden und den Aufwand zur Überführung eines Dokuments in ein Modell bzw. eines Prozessmodells in ein UML-Modell zu verringern, bietet sich die automatische Transformation der Anforderungen in Modelle an. Mit der *Model Driven Architecture* (MDA) definiert die OMG Standards zur Transformation von Modellen in andere Modelle [OMG00], [OMG01], [FeLo03], [Fran03]. Die Transformation von unstrukturierten Dokumenten in Modelle wird von der MDA jedoch nicht abgedeckt. Mit der *Natürlichsprachlichen Informationsbedarfsanalyse* (NIBA) existiert ein Ansatz zur Generierung von UML-Modellen aus Dokumenten, der im Abschnitt 1.2 vorgestellt wird.

1.2 Natürlichsprachliches Parsing als Mittel zur Transformation von Benutzeranforderungen in semiformale Modelle

Das natürlichsprachliche Parsing basiert auf einer grammatikalischen Analyse der Sätze in einem Dokument. Jeder Satz wird in seine einzelnen Worte zerlegt, deren Beziehungen anhand der verwendeten Verben und ihrer Stelligkeit extrahiert werden [FlKM00], [FlMM00].

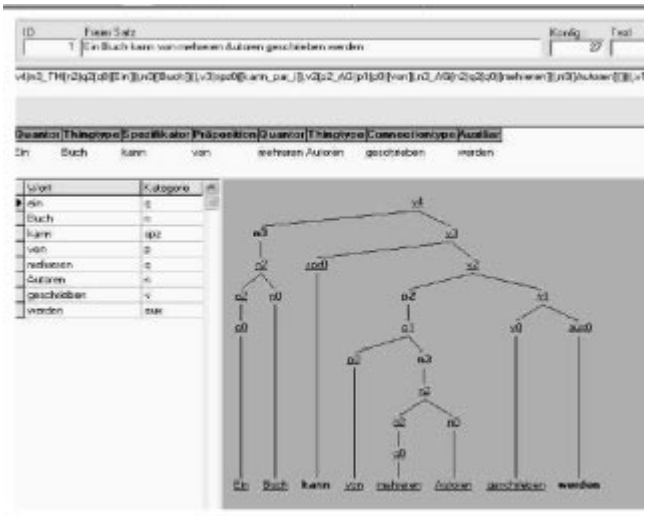

Abbildung 1: Natürlichsprachliches Parsing

Im NIBA-Projekt wurde ein Parser und Mapper implementiert, mit deren Hilfe die entsprechenden Beziehungen gewonnen und in ein UML-Klassenstrukturdiagramm überführt werden können. [FlWe00] Ein Ausschnitt ist in Abbildung 1 dargestellt.

Mangels geeigneter Algorithmen zur generischen Extraktion dieser Beziehungen ist eine Voraussetzung für dieses Verfahren die komplette Abbildung aller deutschen Verben und mit ihren jeweiligen Stelligkeiten. Damit wird das Verfahren bei umfangreichen Texten oder Spezialkontexten sehr aufwendig, wenn das entsprechende Verzeichnis manuell gepflegt werden muss.

In Kapitel 2 werden ein Verfahren zur statistisch-linguistischen Analyse von unstrukturiertem Texten sowie eine Methode zur Überführung von Benutzeranforderungen in UML-Modelle vorgestellt, die in Kapitel 3 zu einer Methode verbunden und in Kapitel 4 auf ein Beispiel angewendet werden.

2. Strukturierung von Benutzeranforderungen

2.1 Patternmatching und statistisch-linguistische Analysen

Der Ansatz des Pattern-Matchings beruht auf einer Matching-Operation `match[pattern, text]`, mit der entweder das Vorhandensein oder alle Positionen eines als Pattern definierten Strings in einen Text oder ein Dokument bestimmt werden [Sun90]. Es existieren verschiedene Verfahren zur Analyse von Texten, wie der Algorithmus von Boyer and Moore, das approximative Pattern-Matching, der Soundex-Algorithmus [Fuh96]. Ergebnis dieser Verfahren sind Begriff-Zitat- oder Begriff–Dokument-Beziehungen.

Als Beispiel für das Pattern-Matching soll der Ansatz der automatischen Gewinnung von Fachbegriffen dienen [Heye02], der in Form des Wortschatztools [Wort] implementiert wurde. Mit Hilfe des Wortschatztools lassen sich aus dem Deutschen Wortschatz[1] die häufigsten Worte, Kookkurrenzen[2] sowie linke und rechte Nachbarn aus einem Text extrahieren und die bestehenden Beziehungen grafisch aufbereiten [Wort]. Diese Operationen werden im folgenden als statistisch-linguistische Analysen verstanden.

Neben den Worten werden im Wortschatztool Zitate angezeigt, in denen die gesuchten Worte vorkommen. Außerdem enthält die zu Grunde liegende Technologie die Möglichkeit zur Identifikation von Aussagen, die im Zusammenhang mit den Worten besonders häufig auftauchen.

---

[1] Im Deutschen Wortschatz werden "sorgfältig ausgewählte öffentlich zugängliche Quellen automatisch erhoben". [Wort, Hinweis unter einem Suchergebnis]

[2] Unter Kookkurrenzen wird das gemeinsame Auftreten von Worten in einem Dokument verstanden. Es wird angenommen, dass die beiden Worte interdependent sind, wenn sie häufig gemeinsam in Dokumenten auftreten. (vgl. http://de.wikipedia.org/Kookkurrenz, [Heye02, S. 4, wobei in der Quelle auf Kollokationen, also das unabhängige gemeinsame Auftreten von Wortformen, abgestellt wird)

Eine Technologie, wie sie im Wortschatztool vorliegt, bildet somit die Grundlage für die Extraktion von Begriffen und Zitaten aus Dokumenten.

2.2 Heuristiken

Von Mario Jeckle wurde 1998 ein Ansatz zur Strukturierung von Anforderungen aus natürlichsprachlichen Dokumenten vorgestellt [Jeck98]. Dieser Ansatz basiert auf der folgenden Heuristik, die auf einen natürlichsprachlichen Fachtext angewendet werden:

1. Mark every noun as candidate class which is not a meaningless expletive or part of a description clarifying some previous expressed information.

2. Check every identified noun (candidate class) if it can receive a value expressible using the basic data types of the intended implementation language. If so the consider expressing the noun as attribute of a class

3. A verb within the textual specification linking two nouns (i. e. concepts) together could be an association within the analysis model. If one of the stated nouns is previously identified as attribute, the verb expresses the implicit interrelation to the containing class. Else all the related nouns were identified as candidate classes, they will be interrelated using a UML association.

Anhand des in [Jeck98] enthaltenen Beispiels, in dem mit Hilfe dieser Heuristik aus einer Prozessbeschreibung ein Klassenmodell entwickelt wird, wird deutlich, dass die Identifikation der Objekt- und Attributkandidaten sowie ihrer Beziehungen intellektuelle Fähigkeiten bei der Anwendung der Heuristik erfordern. Bei Großprojekten wird dieser Ansatz aus Wirtschaftlichkeitsgründen keine Rolle spielen können.

In [GeMa03] wurden Syntaxmustern, Patternorientierter Prozessmodellierung und Sprachnormierung als Heuristik zur Extraktion von Modellinformationen aus natürlichsprachlichen Texten vorgestellt, die mangels Toolunterstützung ebenfalls nur in Modellierungsregeln festgelegt sind und manuell angewendet werden müssen.

3. Kopplung der Verfahren

Die in den vorangegangenen Abschnitten vorgestellten Verfahren zeichnen sich entweder durch hohen Aufwand (NIBA, vgl. 1.2, bzw. Heuristik, vgl. 2.2) oder durch Unvollständigkeit (Wortschatz-Technologie, vgl. 2.1) aus. Koppelt man die Wortschatz-Technologie mit der Heuristik aus [Jeck98] zu einer neuen Methode, eröffnet sich die Möglichkeit zur automatischen Extraktion von kontextspezifischem Fachwissen, das bei der Modellierung von Fachräumen genutzt werden kann. Grundlage des Algorithmus sind Textdokumente D, die der Softwarespezifikation zu Grunde liegen. Diese werden mit Hilfe der statistisch-linguistischen Analyse in die Menge der Worte im Singular W und die Menge der Zitate Z zerlegt. Außerdem wird eine Liste der Trivialworte und Eigennamen T sowie die Liste der attribut- und werttypischen Worte A gebildet.

Trivialworte sind Artikel, Pronomen und Hilfswörter, die für das spätere Modell nicht berücksichtigt werden. Unter attributtypischen Worten sind Worte zu verstehen, die von der Bezeichnung her auf ein Attribut schließen lassen, z. B. ID, Name, Bezeichnung, Größe, Farbe. Werttypische Worte wie true, false, Millionen, DM, Euro, cm bezeichnen typische Attributwerte. Es gilt: $D \supset Z \supset W \supset A$, $W \supset T$.

Um alle Zitate zu finden, die ein bestimmtes Wort $w \in W$ enthalten, wird eine Abbildung Z' definiert: Z': $W \rightarrow Z$: $Z'(w) = \{z \mid w \in z\}$. Das Tupel $b=(w_b, Z'(w_b))$, welches das Wort und seine Zitate enthält, definiert gemäß dem semiotischen Dreieck der Sprachwissenschaft den Begriff b [HeSi80, S. 236]. Die Menge aller Begriffe sei B.

Auf der Menge der Begriffe wird eine Funktion H definiert, die die Häufigkeitsklasse zu jedem Begriff ausgibt [Heye02, S. 3], mit der er im Kontext auftritt. Die häufigsten Begriffe, die keine Trivialworte T darstellen, werden relevante Begriffe B_r genannt. Es gilt: $\forall\, b \in B_r$: $H(b) > h_S \wedge w_b \notin T$. Die signifikante Häufigkeit für Begriffe ist h_S.

Die Beziehungen zwischen Begriffen lassen sich durch die Stärke der Kookkurrenzen bestimmen [Heye02, S. 4]. Für die Kookkurrenz wird eine Funktion Ko': $B_r x B_r \rightarrow IR$ definiert. Die Kookkurrenzen lassen sich als Graph darstellen. Die Knoten bilden die Begriffe, an den Kanten wird die Stärke der Kookkurrenzen vermerkt. Aus Übersichtlichkeitsgründen soll der Graph nur Kookkurrenzen enthalten, deren Wert h_K übersteigt. Deshalb wird die Funktion Ko definiert, die nur signifikant hohe Kookkurrenzen als Funktionswert zurück gibt (relevante Kookkurrenzen), Ko: $B_r x B_r \rightarrow IR$: $Ko(b_1,b_2)=0$, wenn $Ko'(b_1,b_2) \leq h_K$, sonst $Ko'(b_1,b_2)$. Die Menge der relevanten Kookkurrenzen wird mit K_r bezeichnet, $K_r=\{(b_1,b_2) \mid Ko(b_1,b_2)\neq 0\}$ $b1,b2 \in B_r$.

In Anlehnung an die Heuristik in [Jeck98] wird auf die Menge der Begriffe eine Heuristik angewendet. Dazu werden die Menge der UML-Klassen Kl mit $kl_b \in KL$ als der Klasse, die mit dem Begriff b benannt ist, und die Menge der UML-Attribute Att mit $att_b \in Att$, das Attribut mit der Bezeichnung b, gebildet. Die Heuristik lautet:

1. Jeder Begriff, der relevante Kookkurrenzen zu mindestens zwei anderen Begriffen besitzt ($b \in B_r^K$), wird Klassenkandidat. Die Klassifizierungsfunktion ist $Kf: B_r^K \rightarrow Kl$: $b \rightarrow Kf(b) = kl_b$; $B_r^K = \{b \in B_r \mid \exists\, b1, b2 \in B_r: b_1 \neq b_2 \text{ mit } (b, b_1) \in K_r \wedge (b, b_2) \in K_r\}$

2. Jeder Begriff, der relevante Kookkurrenzen zu nur einem anderen Begriff ($b \in B_r^A$) besitzt, wird Attribut. Die dazugehörige Funktion lautet Af: $B_r^A \rightarrow Att$: $b \rightarrow Af(b)=att_b$; $B_r^A = \{b \in B_r \mid \exists!\, b_1 \in B_r \text{ mit } (b, b_1) \in K_r\}$.

3. Ein Begriff zu einem attribut- bzw. werttypischen Wort wird Attribut. $\forall\, b \in B_r \mid w_b \in A$: $b \rightarrow Af(b) = att_b$.

Die vorgestellte Methode hat durch die Kopplung folgende Vorteile. Im Gegensatz zur 2. Heuristik in [Jeck98] (vgl. Abschnitt 2.2) werden die Worte automatisch mit der Liste der attribut- bzw. werttypischen Worte abgeglichen. Während bei [Jeck98, S. 5] die UML-Assoziation mit Hilfe von Verben gebildet wird und somit entweder die Infrastruktur von NIBA (vgl. Abschnitt 1.2) oder intellektuelle Leistungen erfordert, wird im vorgestellten Ansatz die UML-Assoziation mit Hilfe der relevanten Kookkurrenzen gebildet, die automatisch ausgewertet werden können. Modelle können aus natürlichsprachlichen Texten automatisch extrahiert werden. Die Arbeit der Modellentwickler beschränkt somit sich darauf, die Modelle zu prüfen und weiter zu entwickeln.

4. Anwendung und Ausblick

Der vorgestellte Algorithmus soll nun am Beispiel einer Kreditverwaltungssoftware veranschaulicht werden. Für das Wort *Kredit* wird aus dem Wortschatztool die in Abbildung 2 enthaltene Menge der Kookkurrenzen erhoben, die aus Gründen der Einfachheit nur auf Substantive ohne Eigennamen reduziert ist.

Bank (1056), Zins (211), Bürgschaft (112), Sicherheit (100), Hausbank (93), Finanzierung (79)

Abbildung 2: Kookkurrenzen zum Wort "Kredit" ohne attribut- und werttypische Worte

Ermittelt man die Kookkurrenzen zwischen diesen Worten, erhält man bei einem Signifikanzlevel h_K=100 den in Abbildung 3 dargestellten Graph. Die Dicke der Kanten gibt die Stärke der Kookkurrenzen an.

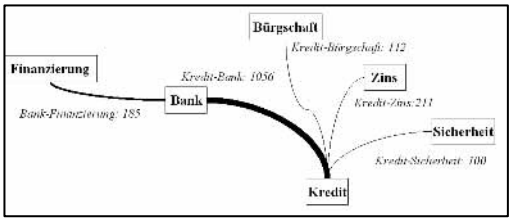

Abbildung 3: Graph über die Kookkurrenzen des Begriffs Kredit

Aus der 1. Heuristik folgt, dass die Begriffe *Kredit* und *Bank* Klassenkandidaten sind. Die Begriffe *Sicherheit, Zins, Bürgschaft* und *Finanzierung* haben nur Beziehungen zu einem anderen Begriff. Sie werden nach der 2. Heuristik Attribute. Somit lässt sich das in Abbildung 4 abgebildete UML-Modell generieren.

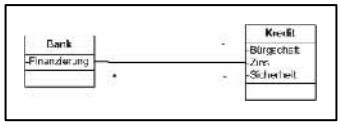

Abbildung 4: Generiertes UML-Modell

Im generierten UML-Klassenmodell wurden die Klassen *Kredit* und *Bank* sowie eine Beziehung zwischen ihnen erkannt. Auch die Attribute der Klasse *Kredit* widerspiegeln kredittypische Eigenschaften. Es wurden somit aus dem vorgegebenen Kontext hilfreiche Informationen zur Konzeption der Kreditverwaltungssoftware gewonnen. Lediglich das Attribut *Finanzierung* der Klasse *Bank* ergibt im Kontext keinen Sinn.

Dass das aus dem Graph generierte UML-Modell einen Fehler aufweist, ist nicht verwunderlich, da als Grundlage für das Modell der Inhalt des Deutschen Wortschatzes, also öffentlich zugängliche Quellen, und keine Spezifikation aus dem entsprechenden Kontext verwendet wurde. An dieser Stelle müsste der Modellentwickler eine Entscheidung über die Verwendung des Wortes *Finanzierung* treffen. Möglich wären die Löschung oder eine eigene (Ober-)Klasse.

Anhand eines Beispiels konnte der Ablauf des Algorithmus demonstriert werden. Es verdeutlicht, dass durch die Kopplung von Verfahren zur statistisch-linguistischen Analyse und einer Heuristik UML-Klassendiagramme aus natürlichsprachlichen Dokumenten erzeugt werden können. Darüber hinaus wird zu zeigen sein, dass dieses Verfahren auch für umfangreiche Dokumente in akzeptabler Zeit Ergebnisse hinreichender Güte liefert. Insbesondere sind die Höhe der Signifikanzwerte h_S und h_K sowie die Gültigkeit der Heuristik für Fachtexte zu prüfen.

Falls diese Überlegungen zu überzeugenden Ergebnissen geführt haben, ist zu untersuchen, inwiefern sich Beziehungen zwischen Klassenkandidaten mit Hilfe der Zitate, in denen sie vorkommen, als Aggregationen oder Spezialisierungen identifizieren lassen. Gleiche Überlegungen sind für Verhaltensdiagramme zu stellen. Die Frage ist, ob aus sich Begriffsbeziehungen (dann auch zu anderen Wortarten, wie Verben und Adjektiven) oder Zitaten allgemeingültige oder kontextspezifische Aussagen zu Prozessabläufen ableiten lassen.

Literaturverzeichnis

[FeLo03] Fettke, P.; Loos, P.: Model Driven Architecture (MDA). In: Wirtschaftsinformatik 45 (2003), Nr. 5, S. 555-559

[FlKM00] Linguistische Aspekte des Projekts NIBA, Klagenfurt 2000.

[FlMM00] Fliedl, G.; Mayerthaler, W/Mayr, Heinrich C. et. al.: The NIBA Workflow – From textual requirements specifications to UML-Schemata, Klagenfurt 2000.

[FlWe00] Fliedl, G.; Werner, G.: NIBA-TAG – A tool for analyzing and preparing german texts, 2000.

[Fran03] Frankel, D. S.: Model Driven Architecture TM: Applying MDATM to Enterprise Computing. 1. Auflage, Indianapolis, Wiley Publishing, Inc. 2003

[Fuh96] Fuhr, N.: Skriptum Information Retrieval, Lecture Notes on Information Retrieval, Universtität Dortmund, 1996

[GeMa03] Gerber, S.; Mai, A.: Modellieren mit Begriffen - ein Vorgehensmodell zur Wiederverwendung von Ergebnissen der Analysephase im Design, in: Petrasch, R.; Wiemers, M.; Kneuper, R. (Hrsg): Praxistauglichkeit von Vorgehensmodellen : 10. Workshop der Fachgruppe WI-VM der Gesellschaft für Informatik e. V. (GI), Aachen, Shaker 2003

[HeSi80] Herberger, M.; Simon, D.r: Wissenschaftstheorie für Juristen : Logik, Semiotik, Erfahrungswissenschaften, Frankfurt am Main, Metzner 1980

[Heye02] Heyer, G.; Quasthoff, U. et al.: Möglichkeiten und Verfahren zur autormatischen Gewinnung von Fachbegriffen aus Texten. In: Bullinger, H.-J. (ed.): Proc. Innovationsforum Content Management –Digitale Inhalte als Bausteine einer vernetzten Welt, Stuttgart 2002.

[Jeck98] Jeckle, M.: On formalising OOA heuristics – Bridging the gap between analysis and design, 1998

[Jec+04a] Jeckle, M.; Rupp, C.; Zengler, B. et. al.: UML 2.0 – Neue Möglichkeiten und alte Probleme, in: Informatik Spektrum, Bd. 27 (2004), Heft 4, S. 323 – 331

[Jec+04b] Jeckle, M.; Rupp, C.; Zengler, B. et. al.: UML 2 glasklar. 1. Auflage. München, Carl Hanser, 2004

[OMG00] OMG: Model Driven Architecture : White Paper Draft 3.2. ftp://ftp.omg.org/pub/docs/omg/00-11-05.pdf 25.01.2005 - OMG

[OMG01] OMG: Model Driven Architecture (MDA). http://www.omg.org/cgi-bin/doc?ormsc/2001-07-01 25.01.2005 - OMG

[Ortn00a] Ortner, E.: Wissensmanagement, Teil 1: Rekonstruktion des Anwendungswissens. In: Informatik Spektrum vom 23.04.2000.

[Ortn00b] Ortner, E.: Wissensmanagement, Teil 2: Systeme und Werkzeuge. In: Informatik Spektrum vom 23.06.2000.

[Part98] Partsch, H.: Requirements-Engineering systematisch: Modellbildung für softwaregestützte Systeme, o. Auflage., Berlin. Springer, 1998.

[Rump05] Rumpe, B.: Agile Modellierung mit UML : Codegenerierung, Testfälle, Refactoring. 1. Auflage. Berlin : Springer, 2005

[Rupp01a] Requirements Engineering und Management, Professionelle, iterative Anforderungsanalyse für die Praxis, Hanser 2001; www.sophist.de

[Rupp01b] Requirements Engineering – der Einsatz einer natürlich-sprachlichen Methode bei der Ermittlung und Qualitätsprüfung von Anforderungen, in: Proceedings der Net.ObjectDays 2001, net.objectdays.org

[Rupp03] Rupp, C.: Wie aus Wünschen Anforderungen werden. In: InformationWeek Nr. 22 vom 30. Oktober 2003, S. 24-25

[RuSc04] Rupp, C.; Schimpfky, N.: Erfahrungsbericht über einen Spezifikationsprozess mit einem bunten Reigen an Methoden und Notationen. In: Softwaretechnik-Trends, Band 24, Heft 4, November 2004

[Sche98] Scheer, A.-W.: ARIS – Vom Geschäftsprozess zum Anwendungssystem, Berlin, Heidelberg, New York, Tokio, Springer 1998.

[Sun90] Sunday, D.: A Very Fast Substring Search Algorithm. Communications of the ACM (CACM), Volume 33, Number 8: S. 132-142 (1990)

[Wiem03] Wiemers, M.: Usability und Vorgehensmodelle. In: Petrasch, R.; Wiemers, M.; Kneuper, R. (Hrsg): Praxistauglichkeit von Vorgehensmodellen – 10. Workshop der Fachgruppe WI-VM der Gesellschaft für Informatik e. V., Aachen, Shaker 2003

[Wort] Wortschatztool der Universität Leipzig, http://www.wortschatz.uni-leipzig.de, zuletzt abgerufen am 27.06.06