Geeignete Repräsentation von Wissen & Regeln in einem vernetzten System

Philipp Dirding
Seminar Innovative Internet-Technologien und Mobilkommunikation, WS 2008/2009
Institut für Informatik, Lehrstuhl Netzarchitekturen und Netzdienste
Technische Universität München
philipp.dirding@mytum.de

Kurzfassung
In diesem Paper soll ein Überblick über die Organisation und den Einsatz von Wissen und Schlussfolgerungen auf diesem Wissen gegeben werden. Zuerst wird ein Einblick in ein Einsatzgebiet gegeben. Hier beschränke ich mich auf das Autonomic Computing, das mit dem von IBM entwickelten Self-Management Konzept erklärt wird.

Anschließend folgt die Definition der eigentlichen Wissensrepräsentation durch Ontologien. Hier soll das theoretische Modell kurz erläutert werden und danach in der Praxis verwendete Konzepte des Modells vorgestellt werden.

Abschließend soll das eAutomation-Konzept von IBM als Referenzmodell vorgestellt werden, als Beispiel für eine Umsetzung von Wissensrepräsentation in der Praxis.

Keywords (Schlüsselworte)
Wissen, Regel, Inferenzregel, Ontologie, Autonomic Computing, Self-Management, eAutomation

1. EINFÜHRUNG
In heutigen Systeme werden sowohl die Hardware als auch die Software immer komplexer[2]. Besonders ERP-Systeme (Enterprise Resource Planning-Systeme) wie SAP ERP oder Oracle E-Business Suite sind nur mit erheblichen Aufwand zu installieren, konfigurieren und betreiben. Aufwand heißt in diesem Zusammenhang, dass hohe Kosten durch qualifizierte IT-Experten entstehen und der Betrieb viel Zeit verschlingt.

Aber nicht nur einzelne Anwendungen sind komplex, sondern auch die Integration in eine Umgebung mit verschiedenen unterschiedlichen Systemen[2]. Gerade diese Integration wird in Zeiten des Pervasive Computing, in denen mehr und mehr Informationstechnologien im Alltag eingesetzt werden und zusammenarbeiten müssen, immer wichtiger und umfasst immer mehr Systeme[2].

Diese Komplexität der einzelnen Systeme und der Systemlandschaften ist in der Vergangenheit immer größer geworden und wird auch in Zukunft weiter wachsen.

Um den Problemen, wie hoher Installations-, Konfigurations- und Administrationsaufwand, bei diesen komplexen Systemen zu begegnen, wurde, besonders angetrieben von IBM, das Konzept des Autonomic Computing entwickelt[8].

Autonomic Computing hat sich zum Ziel gesetzt, die Administratoren von Detailfragen und Routineaufgaben zu befreien und ihre Arbeit auf die Gestaltung von groben Vorgaben für das System zu beschränken. Mit diesen Vorgaben sollen sich die Systeme selbst organisieren und an ihre Umgebung anpassen, damit sie ihre Aufgaben möglichst gut erfüllen.

Autonomic Computing wird in [2] verglichen mit dem Nervensystem verglichen, das Herzfrequenz und Körpertemperatur regelt, ohne dass das Gehirn bewusst in diese Prozesse eingreifen muss.

1.1 SELF-CHOP
Autonomic Computing umfasst mehrere Teilbereiche. IBM teilt sie in vier verschiedene Aufgaben auf: Self-configuration, Self-healing, Self-optimization und Self-protection. Abgekürzt werden diese vier Punkte nach ihren Anfangsbuchstaben als Self-CHOP[1].

1.1.1 Self-configuration
Unter Self-configuration versteht man die autonome Installation, Konfiguration und Integration in die Umgebung, großer komplexer Systeme. Diese Prozesse können, von IT-Experten durchgeführt, in einer heterogenen Hard- und Softwarelandschaft sehr zeitaufwendig und kostspielig sein. Autonome Systeme erledigen die Aufgaben selber nach Zielvorgaben auf einer höheren Ebene (high level policies). Diese Zielvorgaben beschreiben Wunschzustände, aber nicht wie diese erreicht werden[2].

In Self-configuration-Umgebung müssen Systeme nur noch in diese eingebracht werden und danach erkennt das System die Umgebung eigenständig und installiert und konfiguriert sich entsprechend der anderen Systeme in der Umgebung und ebenso erkennen die anderen Systeme die neue Komponente und reagieren auf diese entsprechend um die Zielvorgaben zu erfüllen[2].

1.1.2 Self-healing
Systeme erfüllen das Konzept Self-healing wenn sie eigenes Fehlverhalten autonom erkennen, verfolgen und die Ursache bestimmen. Als Fehlverhalten gelten hier Bugs oder Ausfälle in Hard- und Software. Die Diagnose kann über verschiedene Arten von Detektoren geschehen. So können Sensoren gewisse Parameter überwachen oder Logdateien analysiert werden und mit dem Sollstatus des Systems verglichen werden[2].

Dann soll die gestellte Diagnose mit Fehlerbehebungsroutinen verglichen werden, zum Beispiel Softwarepatches und diese dann angewendet und nochmals getestet werden[2]. Alternativ wird das Problem an die Administratoren weitergegeben.

Self-healing-Funktionen übernehmen damit die Arbeit der großen Abteilungen, die für die Identifikation, Verfolgung und Ursachenfindung von Fehlern zuständig sind und für die Arbeit oft Wochen brauchen und nicht immer zu Lösungen kommen[2].

1.1.3 Self-optimization
In komplexen Systemen gibt es sehr viele verschiedene Stellschrauben für Performanz[2]. Menschen können diese erstens nicht alle überblicken und zweitens ihre Auswirkungen oft nicht

abschätzen, insbesondere im Zusammenhang mit anderen integrierten Komponenten.

Unter Self-optimization versteht man die Verlagerung der Aufgabe das System möglichst performant einzustellen auf das System selber. Durch kontinuierliche Überwachung und Experimentieren mit den Möglichkeiten die Performanz zu erhöhen, lernt das System immer genauer die Entscheidungen zu treffen, die zu einer erhöhten Performanz und Effizienz führen.

1.1.4 Self-protection

Unter self-protection-Systemen versteht Systeme, die sich selbst vor Angriffen von außerhalb des Systems und vor Verkettungen von Fehlern, die nicht durch self-healing-Funktionen behoben werden konnten. Diese Angriffe und Fehler werden nicht nur bei akutem Auftreten erkennt und beseitigt werden, sondern das System soll diese durch Analyse verschiedener Sensoren antizipieren können und sich dagegen schützen[2].

2. WISSENSREPRÄSENTATIONEN

Autonomic Computing-Systeme handeln somit zu einem gewissen Grad, wie der Name schon sagt, autonom nach Zielvorgaben. Es liegen nicht wie bei automatischen Systemen genaue Handlungsanweisungen vor, nach denen das System agiert, sondern Entscheidungen muss das System eigenständig treffen. Diese Entscheidungen treffen Systeme auf der Basis einer Wissensgrundlage. Regeln, die aus den Zielvorgaben auf höherer Ebene generiert werden, werden auf dieses Wissen angewendet. Dieses Anwenden der Regeln auf das Wissen ermöglicht den autonomen Systemen Schluss zu folgern (to reason). Das Ergebnis dieser Schlussfolgerungen ist wiederum neues Wissen oder Handlungsanweisungen.

Welches Wissen das genau sein kann, hängt von den einzelnen Modellen und Implementierungen ab, aber das Wissen muss im System irgendwie repräsentiert werden. Eine Möglichkeit Wissen zu repräsentieren sind Ontologien.

2.1 Ontologie

Ontologien sind formale Spezifikationen, für die Strukturierung von Wissen. Zwei Arten von Regeln bestimmen diese Struktur. Integritäts und Inferenzregeln[9].

Die Integritätsregeln, sind Regeln die den Aufbau des Wissens bestimmen. Sie bestimmen zum Beispiel welche Beziehungen zwischen Wissensentitäten erlaubt sind oder wie einzelne Wissensentitäten aussehen können.

Die Inferenzregeln ermöglichen das Schlussfolgern auf der Basis des vorhandenen Wissens. Aus dem vorhandenen Wissen lassen sich Schlüsse ziehen und somit neues Wissen generieren. Ist zum Beispiel bereits das Wissen vorhanden, dass sich zwei Server A und B im selben Raum befinden, und dazu bekannt wird, dass B und C im selben Raum stehen, lässt sich schlussfolgern, dass A und C auch im selben Raum stehen.

Da sowohl Mensch als auch Maschine diese Regeln kennen, nach denen das Wissen aufgebaut ist, ermöglichen Ontologien die Kommunikation zwischen Mensch und Maschine.

Ontologien sind immer auf einen bestimmten Einsatzzweck spezialisiert, der eine bestimmte Wissenstruktur erfordert und daher gibt es viele verschiedene Ontologien, die sich mehr oder minder stark unterscheiden[4]. Es wäre unpraktisch, wenn nicht sogar unmöglich alles vorhandene Wissen in einer Ontologie unterbringen zu wollen. Nun gibt es mehrere Wege, das Wissen strukturiert sein kann, d.h. die Integritätsregeln aussehen. Im Folgenden sollen einige gebräuchliche Konzepte vorgestellt werden.

2.2 WISSENSTRUKTUREN

2.2.1 Taxonomie

Einer der einfachsten Ansätze Wissen zu strukturieren ist eine Taxonomie. Eine Taxonomie teilt Gegenstände, genannt Entitäten, hierarchisch in Kategorien ein.

Erfunden wurde die Taxonomie von dem Biologen Carl von Linné zur Einteilung aller Lebensformen in Familien, Gattungen, Arten[4].

Taxonomien bilden Über- und Unterordnungsbeziehungen zwischen den Entitäten ab und können somit auch Vererbung darstellen.

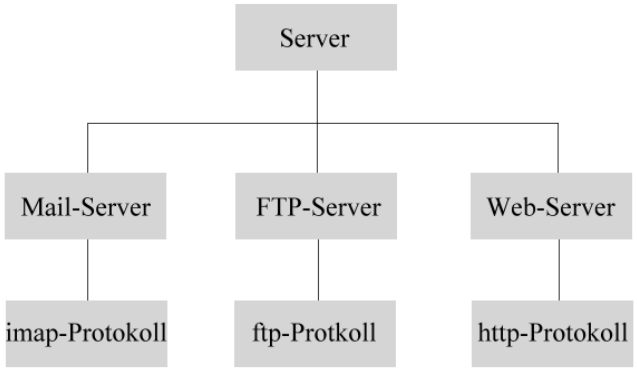

Abbildung 1. Eine einfache Taxonomie.

Beziehungen zwischen Kategorien einer Ebene sind nicht vorgesehen.

2.2.2 Thesaurus

Ein Thesaurus strukturiert ebenfalls Wissen ist ein Schlagwortverzeichnis oder kontrolliertes Vokabular, dessen Begriffe untereinander in Beziehung stehen.

Zu einem Begriff werden zum Beispiel Ober- und Unterbegriffe, Synonyme, Homonyme oder Äquivalenzbeziehungen gespeichert. Und jeder dieser Begriffe hat selber wieder Ober- und Unterbegriffe usw. So lassen sich Themenbereiche genau beschreiben und repräsentieren.

Für Thesauri gibt es verschiedene Normen, die zum Beispiel die verschiedenen Arten der Relationen vorgeben.

LAN

-Synonyme
  Local-Area Networks
  Ethernet
  Inhouse-Netz
  Local Area Network
  Lokales Netz
-Oberbegriffe
  Computernetz
-Verwandte Begriffe
  Intranet
  Unternehmensnetzwerk
-Zuordnung
  N.10.07.01 Informationstechnik und Systemarchitektur
  B.09.02 IS-Entwicklung und -Betrieb

Abbildung 2. Thesaurus LAN[11]

2.2.3 Semantisches Netz

Ein semantisches Netz ist ein graphisches Modell, in dem Begriffe untereinander in Beziehung gesetzt werden. Begriffe werden durch Knoten dargestellt, während die Kanten zwischen den Knoten die inhaltlichen Beziehungen der Begriffe untereinander repräsentieren.

Es gibt verschiedene semantische Netze, die unterschiedliche Beziehungstypen erlauben. Die vorher vorgestellten Wissensstrukturen Taxonomie und Thesaurus sind semantische Netze mit einer begrenzten Anzahl von Relationen[7].

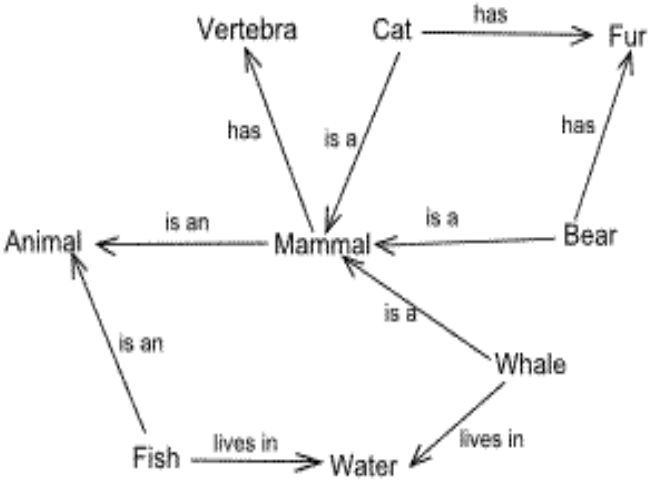

Abbildung 3. Beispiel eines semantischen Netzes[7]

2.2.4 Frames

Frames orientieren sich an der Idee, dass der Mensch in gewissen Schemen, also Abstraktionen von der Realität, denkt und sollen den Rahmen für diese Schemen bilden[4].

Das Konzept erinnert stark an das Objekt-orientierte Klassen-Modell, allerdings fehlen im die Methoden. Außerdem ist es eng verwandt mit dem semantischen Netz.

Jede Wissensentität ist ein Frame und hat einen Namen und besitzt Attribute, die Slots genannt werden. Diese Slots können sowohl eine Eigenschaften des jeweiligen Frames sein, als auch eine Beziehung zu einem anderen Frame darstellen[4].

Ausgehend von der menschlichen Idee eines Büros kann man ein Frame konstruieren:

Der Name des Frames ist „Büro“. In einem Büro erwartet man einen Schreibtisch, einen Bürostuhl und Fenster. Diese drei Dinge sind Slots und stellen Beziehungen zu jeweils einem anderen Frame dar. Der Schreibtisch ist wieder ein Frame, das Slots (z.B. Material) hat. Ein Slot der eine Eigenschaft des Frames „Büro“ ist, könnte zum Beispiel die Größe in Quadratmetern sein.

So bildet sich ein „Netz“ aus Gegenstände oder Wissensentitäten in Form von Frames.

2.2.5 Prädikatenlogik

Die Aussagenlogik ist die Grundlage für die Prädikatenlogik. Sie besteht aus Prädikaten und Termen. So kann der Satz „Die Sonne scheint“ durch ein Prädikat und einen Term dargestellt werden:

scheint(Sonne)[5]

Prädikate stellen die Namen von Eigenschaften, Relationen oder Klassen dar, während Terme Objekte des Diskursbereichs darstellen[6].

So lassen sich Relationen zwischen verschiedenen Termen bilden, wenn sie zum Beispiel ein Prädikat teilen.

Es ergibt sich allerdings ein Problem bei der Prädikatenlogik. Die Folgerungen geschehen ohne inhaltliche Wertung und somit gibt es einen Satz „erlaubter“ Lösungen, aber man keine dieser Lösungen kann „empfohlen“ werden[4].

Prädikatenlogik ist jedoch einfach in Software zu implementieren, da mit Prolog eine logische Programmiersprache existiert die auf diesem Modell basiert.

3. REFERENZMODELL: eAutomation

Da vor allem das Self-Management von Systemen stark von ihnen vorangetrieben wird, hat IBM ein Referenzmodell für dieses Konzept entwickelt, genannt eAutomation. Dieses Referenzmodell ist zum Beispiel in die IBM Tivoli System Automation für z/OS eingeflossen.

3.1 Correlation engine

eAutomation ist eine correlation engine mit der sich drei Analysetypen durchführen lassen[3]:

- Data filtering
- Thresholding
- Sequencing

Data filtering ist zum Beispiel die Unterdrückung multipler Fehlermeldungen, die ein und dieselbe Ursache haben. Produziert zum Beispiel ein Netzwerkswitch Fehler, so sind bestimmte Server nicht mehr erreichbar. Hier werden vom Netzwerkswitch, von den Servern, die nicht mehr erreichbar sind, und von den Komponenten, die auf den Server zugreifen wollen Fehlerberichte geschickt, obwohl der Fehler mit der Benachrichtigung vom Switch umfassend beschrieben ist[3].

Thresholding beschreibt Grenzwerte. Über Thresholding kann zum Beispiel beschrieben werden, dass wenn eine Aktion x Mal fehlschlägt, ein ernsthafter Fehler aufgetreten ist[3].

Sequencing alarmiert zum Beispiel, wenn eine Sicherheitstür mehr als eine bestimmte Zeit offen steht[3].

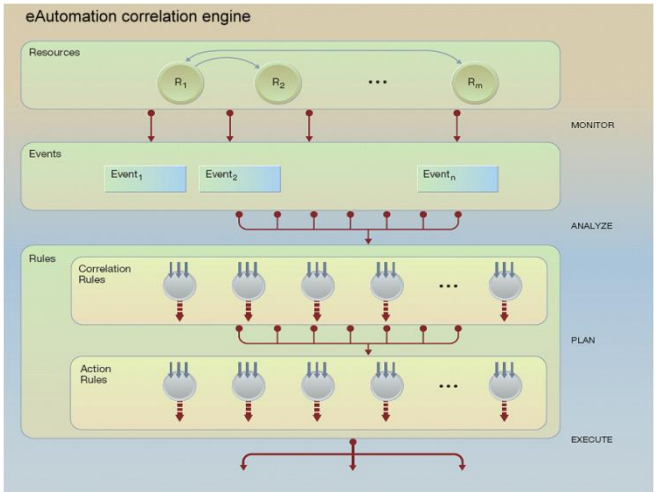

Abbildung 4. Die eAutomation correlation engine[3]

Die correlation engine in der Lage erstens auf Probleme bzw. Fehler zu reagieren und zweitens kann sie Probleme und Fehler antizipieren um sie zu vermeiden.

3.2 MAPE-Modell

Das MAPE-Modell beschreibt die Gesamtarchitektur des Self-Management-Systems und bricht dieses in vier einzelne Teile:

- Monitor
- Analyze
- Plan
- Execute

Diese vier Teile sind im Grunde Teil eines jeden Managementprozesses. Zuerst werden vorhandene Daten gesammelt (Monitor). Diese gesammelten Daten werden analysiert (Analyze). Aus den gewonnen Erkenntnissen werden Aktionen vorbereitet (Plan) und ausgeführt (Execute).

Da alle vier Schritte auf demselben Problembereich arbeiten, muss sichergestellt werden, dass das geteilte Wissen von allen verstanden wird.

Diese Integration wird durch eine Aufteilung der correlation engine auf drei Ebenen[3]:

- Resource layer
- Event layer
- Rule layer

Diese Aufteilung korreliert mit dem MAPE-Modell wie man in Abbildung 4 entnehmen kann.

Resourcen können quasi alles sein, aber sie stellen immer etwas dar, was vom Self-management-System verwaltet werden soll. Das kann ein Server, ein bestimmtes Programm oder auch ein Service sein.

Events sind nichts anderes als Statusänderungen von Ressourcen (z.B. Ausfall eines Webservers) und durch Weitergabe dienen sie als Benachrichtigungen.

Diese Benachrichtigungen werden vom rule layer aufgenommen.

3.3 Ontologie

Das eAutomation-Modell besitzt eine eigene Ontologie, die auf der KOANA-Ontologie der Uni Karlsruhe basiert, welche wiederum auf semantischen Netzen basiert.

3.3.1 Resourcen

Das Kernelement der eAutomation-Ontologie ist die Ressource, da diese das eigentliche zu managende Objekt darstellt. Das Modell, wie es in der Ontologie realisiert wurde ist in Figure-3 dargestellt.

Jede Ressource hat einen eindeutigen Namen. Der Status der einer Ressource und damit zwei weitere sehr wichtige Attribute sind Current_Operational_State und Desired_Operational_State[3]. Ersteres beschreibt den aktuellen Zustand der Ressource. Jede Änderung dieses Attributs ist ein Event. Zweiteres Attribut spezifiziert den gewünschten Zustand der Ressource. Werte für diese Attribute sind nicht frei wählbar sondern vordefiniert. So kann für einen Server zum Beispiel gelten, dass er für die beiden genannten Attribute „online“ und „offline“ annehmen kann.

Ressourcen untereinander können zwei Zusammenhänge haben. Es gibt start/stop-Beziehungen und location-Relationen. Eine

Start/stop-Relation ist zum Beispiel *startAfter*, die angibt, dass für den Start einer Ressource die in startAfter enthaltene, den Current_Operational_Status „online“ haben muss. Location-Relationen sind zum Beispiel *Collocated* und *AntiCollocated*. *Collocated* gibt an, dass eine Ressource nur da in Betrieb sein darf wo die entsprechend andere vorhanden ist. *AntiCollocated* beschreibt genau das Gegenteil.

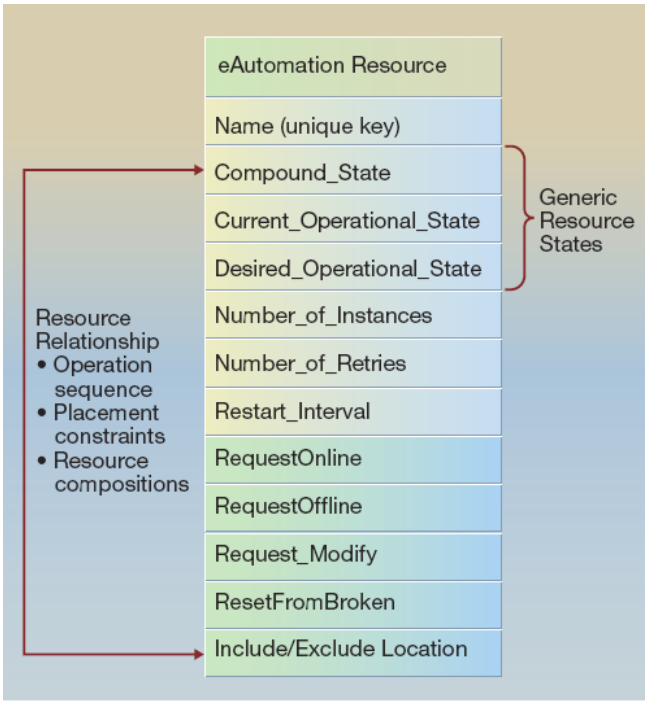

Abbildung 5. Modell einer Ressource[3]

3.3.2 Regeln

Die eAutomation correlation engine unterstützt zwei Arten von Regelsätzen: correlation und action rules.

Correlation rules können zum Beispiel Zusammenhänge zwischen Events erkennen und diese Zusammenfassen und mit einer sinnvollen Meldung versehen (data filtering). Sie erleichtern dem Administrator die Problemsuche oder das Verständnis des Problems.

Action rules dagegen sind Regeln, die basierend auf Events aktiv im System Veränderungen erzeugen können. Das nimmt dem Administrator tatsächliche Arbeit ab.

4. Literatur

[1] Miller, B. (2005, September). The autonomic computing edge: Can you chop up autonomic computing. http://www.ibm.com/developerworks/library/ac-edge4/index.html (16.11.2008)

[2] Kephart, J. O. and D. M. Chess (2003). The vision of autonomic computing. Computer 36 (1), 41-50.

[3] Stojanovic, L., J. Schneider, A. Maedche, S. Libischer, R. Studer, T. Lumpp, A. Abecker, G. Breiter, and J. Dinger (2004). The role of ontologies in autonomic computing systems. IBM Systems Journal 43 (3).

[4] Davis, R., H. Shrobe, and P. Szolovits (1993). What is a knowledge representation? AI Magazine 14 (1), 17-33.

[5] o.V. Prädikatenlogik. http://www.ifi.uzh.ch/req/courses/logische_programmierung/ws03/documents/Praedikatenlogik.pdf (12.11.2008)

[6] François Bry. (2004). Wie können Daten aus dem Web automatisch gefunden werden? Aspekte eines aktuellen Informatikvorhabens

http://www.pms.ifi.lmu.de/mitarbeiter/bry/tag-der-mathematik-2004/tag-der-mathematik-2004.html (13.11.2008)

[7] http://en.wikipedia.org/wiki/Semantic_net (18.11.2008)

[8] http://de.wikipedia.org/wiki/Autonomic_Computing (28.01.2009)

[9] http://de.wikipedia.org/wiki/Ontologie_(Informatik) (28.01.2009)

[10] http://de.wikipedia.org/wiki/Wissensrepr%C3%A4sentation(28.01.2009)

[11] http://www.genios.de/thesaurus/subthesaurus/deskriptor/desk 2827.html