# Systematische Berücksichtigung von Abhängigkeitsbeziehungen bei Architekturentscheidungen

Sven Wohlfarth, Matthias Riebisch

TU Ilmenau, FG Prozessinformatik/Softwaresysteme, 98693 Ilmenau
{sven.wohlfarth|matthias.riebisch}@tu-ilmenau.de

**Abstract:** Architekturentscheidungen gehören zu den riskantesten Entscheidungen in großen Entwicklungsprojekten, da häufig nur unvollständige und unstrukturierte Informationen vorliegen. Gerade aufgrund der lückenhaft dokumentierten Abhängigkeitsbeziehungen steigen die mit einer Entscheidung verbundenen Risiken. Da ein Verfahren zur Erkennung und Berücksichtigung der Abhängigkeitsbeziehungen bei Architekturentscheidungen bislang fehlt, ist ein geeigneter Entscheidungsprozess zu entwickeln. Das Ziel ist es, die komplexen Abhängigkeitsbeziehungen trotz unvollständiger Informationsgrundlage sichtbar zu machen, im Detail zu analysieren und sie systematisch bei der Entscheidungsfindung zu berücksichtigen.

## 1 Notwendigkeit der Systematisierung der Entscheidungsfindung

In langfristigen Großprojekten sind Architektur- und Reengineering-Entscheidungen schwer zu treffen und mit erheblichen Risiken für den Auftraggeber verbunden, da Abhängigkeitsbeziehungen nur lückenhaft bekannt sind. Es ist daher ein Verfahren erforderlich, um die Abhängigkeitsbeziehungen sichtbar zu machen und sie systematisch und in der erforderlichen Tiefe bei der Entscheidungsfindung berücksichtigen zu können. Architekturentscheidungen in Großprojekten erfolgen unter speziellen Bedingungen, wie einer hohen Zahl von Entwicklern, räumlich getrennten Teilprojekten und paralleler Entwicklung von getrennten Versionen. Ein hoher Termindruck und eine Vielzahl an Veränderungen erschweren die Aufrechterhaltung einer vollständigen und konsistenten Dokumentation zur Ermittlung der Abhängigkeitsbeziehungen. Das Risiko von Fehlentscheidungen für das betroffene Unternehmen besteht in hohem Korrekturaufwand sowie hohen finanziellen Verlusten bei Ausfall eines geschäftskritischen Systems.

Ein strukturiertes, systematisches Vorgehen bei der Entscheidungsfindung hilft, die Komplexität zu beherrschen und Fehler zu vermeiden. Im ersten Schritt eines solchen Vorgehens werden die Ziele und Rahmenbedingungen erhoben und strukturiert. Dabei sind Widersprüche und Konkurrenz-Beziehungen zwischen den Zielen aufzulösen. Danach werden vom Architekten Lösungsansätze zur Veränderung der Architektur zwecks Erfüllung der Ziele entwickelt. Im Fall existierender Komponenten oder Softwaresysteme sind diese auf Abhängigkeitsbeziehungen zwischen den Komponenten hin zu analysieren. Dann sind die alternativen Lösungsansätze anhand der erwarteten Beiträge zur Zielerreichung und möglicher Seiteneffekte auf andere Systemteile zu bewerten, um eine Entscheidung nach rationalen Gesichtspunkten treffen zu können. Diese Schritte (siehe

die Zusammenfassung in Abbildung 1) folgen den Vorschlägen des Vorgehens zur Entscheidungsfindung aus der präskriptiven Entscheidungstheorie [EW03].

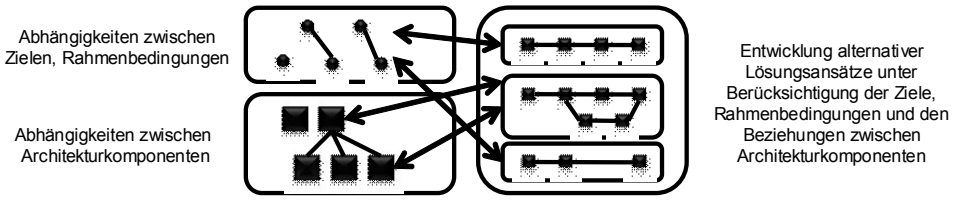


Abbildung 1: Berücksichtigung der Abhängigkeitsbeziehungen bei der Entscheidungsfindung

## 2 Typen von Abhängigkeitsbeziehungen

Abhängigkeitsbeziehungen spielen für das Verständnis von Zusammenhängen durch Architekten und Entwickler eine besonders wichtige Rolle. Sie werden für Aufstellung, Bewertung und Prüfung von Lösungsansätzen benötigt. Zu ihrer Modellierung und Beschreibung in verschiedenen Methoden der Softwaretechnik wurde eine Vielzahl verschiedener organisatorischen und technischen Abhängigkeitsbeziehungen verwendet, die teilweise unter verschiedenen Bezeichnungen geführt werden. Für ein einheitliches Verständnis wurde aus dieser Menge in Anlehnung an [KL05] eine für die Entscheidungsfindung relevante Auswahl von Typen getroffen (Tabelle 1). Eine Fokussierung auf wenige Beziehungstypen ist notwendig, um Abhängigkeiten besser analysieren und ihren Einfluss auf Entscheidungen bewerten zu können.

Tabelle 1: Typen von Abhängigkeitsbeziehungen

| ***Beziehungen zwischen*** | **Komponenten** | **Maßn. zur Architektur-Veränderung** | **Ziele, Rahmenbedingungen** |
|---|---|---|---|
| **Komponenten** | ‚bedingt durch'<br>‚benutzt von'<br>‚aggregiert mit'<br>‚spezialisiert zu' | ‚Teil von' | ‚in Konflikt mit'<br>‚erfüllen vollständig/ teilweise' |
| **Maßnahmen zur Architektur-Veränderung** | | ‚erzwingen'<br>‚in Konflikt mit'<br>‚zusammengefasst' | ‚beeinflusst von' |
| **Ziele, Rahmenbedingungen** | | | ‚sich ergänzend'<br>‚miteinander konkurrierend' |

Die in Tabelle 1 aufgeführten Beziehungstypen beziehen sich auf folgende Elemente von Architekturentscheidungen (siehe Abbildung 1):

- **Ziele und Rahmenbedingungen** können sich ‚ergänzen', in einer Ziel-Mittel-Beziehung stehen, ‚konkurrieren' und sich gegenseitig ausschließen. In diesen Fällen muss ein Kompromiss gefunden werden. Beispiele: Budget-Restriktionen, Vorgaben hinsichtlich der zu verwendenden Frameworks.
- **Architekturkomponenten** können sich gegenseitig ‚benutzen' oder in einer ‚aggregiert mit' / ‚spezialisiert zu' Beziehung stehen. Diese Beziehungen geben wich-

tige Hinweise darauf, welche Komponenten unabhängig und welche Komponenten ausschließlich in Kombination verändert, ausgetauscht oder ergänzt werden können. Beispiele: Softwarekomponenten wie Broker, Event-Handler oder Datenbanken.

- **Maßnahmen zur Veränderung der Architektur** können miteinander ‚in Konflikt' stehen oder sich gegenseitig ‚erzwingen'. Gerade umfassende Architekturveränderungen müssen aus Risiko-, Ressourcen- und Steuerungsaspekten heraus in kleinen Schritten durchgeführt werden. Da zum Zeitpunkt der Entscheidung eine Sequenz oder eine Folge mehrerer Maßnahmen gleichzeitig zu berücksichtigen ist, ist die Kompatibilität der Maßnahmen zueinander sicherzustellen. Beispiele sind der Austausch, die Zusammenfassung oder die Teilung einer Komponente.

## 3 Systematische Berücksichtigung der Abhängigkeitsbeziehungen bei der Entscheidungsfindung

Um die in Tabelle 1 dargestellten Beziehungen bei der Entscheidungsfindung zu berücksichtigen sind zunächst die Beziehungen zwischen den **Zielen und Rahmenbedingungen** und zwischen den **Architekturkomponenten** zu identifizieren.

Zur Illustration werden hier Ausschnitte eines größeren Projektes angeführt, die eine Architektur-Veränderung am Web-Content-Management-System Typo3 behandeln. Typo3 ist eine Open Source PHP-Webapplikation zur Gestaltung und Konfiguration von komplexen Webseiten. Typo3 ist modular aufgebaut; neben einem Typo3-Kern existieren über 3.000 sogenannte Extensions, die einen Einsatz von Typo3 in vielen Anwendungsdomänen ermöglichen [FR05].

**Widerspruchsfreie Strukturierung der Ziele und Rahmenbedingungen**

Zur Strukturierung ‚konkurrierender' oder ‚sich ergänzender' Ziele wird in der ersten Phase ein Ziel-Mittel-Diagramm erstellt. Das Vorgehen hierzu entstammt aus der Entscheidungstheorie [EW03]. Dabei werden die Ziele zerlegt, verfeinert, nach Instrumental- und Fundamentalzielen klassifiziert und in eine baumartige Ziel-Mittel-Struktur eingeordnet. An der Spitze stehen die Fundamentalziele, welche durch die verschiedenen Instrumentalziele verfeinert werden. Diese Art der Strukturierung der Ziele erleichtert die anschließende systematische Priorisierung der Ziele. Die Zerlegung wird fortgesetzt, bis jedes Ziel in eindeutigen Ziel-Mittel-Beziehungen zu den anderen Zielen steht.

Im Beispiel Typo3 besteht das Ziel in einer Steigerung der Portabilität bezüglich des verwendeten Datenbank-Managementsystem DBMS. Ein zentraler Schwachpunkt von Version 3.6 ist die Bindung an das DBMS mySQL, was die Verwendung in einem Unternehmensumfeld stark einschränkt, weil hier häufig andere DBMS etabliert sind. Um als Fundamentalziel die Portabilität in der Version 4 zu verbessern, sind geeignete Lösungsansätze zur Restrukturierung des Datenbankzugriffs zu entwickeln und zu vergleichen (siehe Abbildung 2). Ein konkurrierendes Fundamentalziel sind die Antwortzeitverhalten. Da kurze Antwortzeiten für Typo3 ein kritischer Erfolgsfaktor sind, darf die Restrukturierung keinesfalls zu einer Verschlechterung führen. Eine Beeinträchtigung der Antwortzeiten ein hohes Risiko dar, da ein Wechsel zu einem anderen Content-Management-System mit erheblichem Aufwand verbunden wäre.

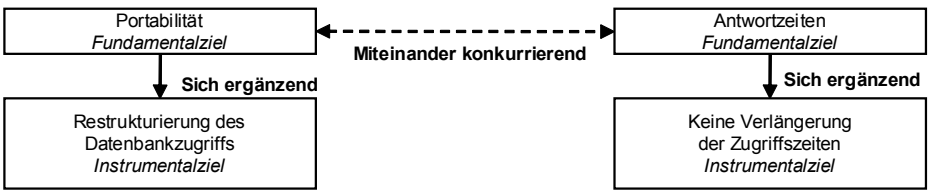


Abbildung 2: Erstellung eines widerspruchsfreien Ziel-Mittel-Diagramms

Um die Identifizierung und Strukturierung der Rahmenbedingungen, z.B. die Ermittlung des Budgets oder der technischen Bedingungen der Implementierungsplattform, systematisch durchzuführen, kann eine Checkliste eingesetzt werden. Sie umfasst einerseits die Rahmenbedingungen tabellarisch in gruppierter Form und andererseits die zugehörigen absoluten Werte oder Wertebereiche. Mit der Abarbeitung der Checkliste werden die Rahmenbedingungen zunächst vollständig erhoben; sich ergänzende Rahmenbedingungen werden dabei hervorgehoben. Die Auflösung von Konkurrenzbeziehungen erfolgt im zweiten Schritt durch eine Priorisierung der Rahmenbedingungen im Hinblick auf die Zielerreichung und die Anpassung der Rahmenbedingungen mit der geringeren Priorität.

**Ermittlung der Abhängigkeitsbeziehungen zwischen Architekturkomponenten**

Um die häufig nur lückenhaft bekannten Beziehungen zwischen den Architekturkomponente identifizieren zu können, ist eine Beschreibung der relevanten Teile der Architektur erforderlich. Um den Aufwand für die Beschreibung zu begrenzen, ist eine Abgrenzung des Betrachtungsraums anhand der Ziele und ggf. bereits bekannter Schwachstellen notwendig. Die Beschreibung des relevanten Teils als Modell erfolgt mittels üblicher Architektur-Beschreibungssprachen. Um die Abhängigkeitsbeziehungen zu ermitteln, wird ein Architektur-Review mittels einer szenariobasierte Architekturanalyse durchgeführt. Als Grundlage kann hierfür die Architecture Tradeoff Analysis Method ATAM [KK00] angewendet werden. Je nach Ziel werden Szenarien auf das Modell der Softwarearchitektur angewendet, um die Abhängigkeiten zwischen Komponenten erkennen zu können (siehe Abbildung 3). Zur Bewertung des Veränderungsaufwands kann die Architecture Level Modifiability Analysis ALMA [BL04] verwendet werden, die geeignete Szenarien für ATAM definiert. Sofern der Betrachtungsraum groß ist, können zur

Identifizierung von Schwachstellen zunächst Architekturmetriken ausgewertet werden, z.B. das Distanzmaß [SL99] oder Kopplungs-/Kohäsionsmetriken [SA01].

Beim Beispiel Typo3 mit seiner modularen Architektur aus einem Kern ‚Typo3-Core' und der Erweiterung durch ‚Extensions'(siehe Abbildung 3) ist eine Vielzahl von Abhängigkeitsbeziehungen zu berücksichtigen, da alle ‚Extensions' auf die Datenbank zugreifen. Der Fokus liegt jedoch auf den Beziehungen zwischen ‚Typo3-Core' und dem ‚Database-Abstraction-Layer', da die Programmierrichtlinien die Nutzung des Layers für den Zugriff auf die Datenbank vorschreiben. Dieser Layer stellt eine Vielzahl von Funktionen zur Verfügung, um Daten aus einer MySQL-Datenbank abzurufen, zu manipulieren und um neue Daten einzustellen. Bei der Restrukturierung müssen diese Funktionen in identischer Art und Weise benutzt werden können, um die Kompatibilität einer Vielzahl von ‚Extensions' zu gewährleisten.

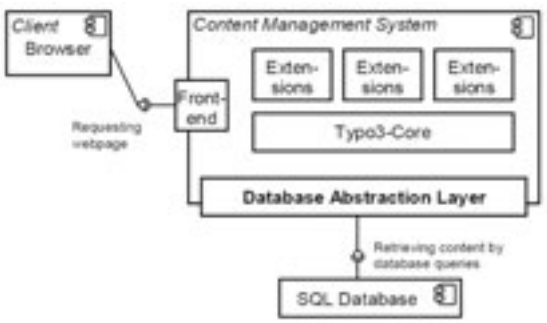


Abbildung 3: Modulare Architektur von Typo3 als UML-Komponentendiagramm

**Systematische Entwicklung und Verfeinerung alternativer Lösungsansätze**

In der dritten Phase erfolgt die Entwicklung und Verfeinerung der alternativen Lösungsansätze, mit denen die Architektur zur Erfüllung der Ziele verändert wird. Hierfür sind vom Architekten zunächst geeignete Lösungsansätze wie Best Practices, Muster, Stile und ähnliches auszuwählen. Zur Bewertung werden je alternativem Lösungsansatz die notwendigen Maßnahmen zur Architektur-Veränderung identifiziert. Dabei wird eine Reihenfolge der Maßnahmen festgelegt, wobei Abhängigkeiten zwischen den Maßnahmen berücksichtigt werden. Im Falle von Konflikten wie beispielsweise fehlenden Voraussetzungen muss die Reihenfolge verändert oder es müssen zusätzliche Maßnahmen eingefügt werden. Im Beispiel der der Typo3-Architekturveränderung muss vor Beginn der Restrukturierungen sichergestellt werden, dass alle Datenbankzugriffe über den ‚Database-Abstraction-Layer' erfolgen.

Zur Ermittlung der Reihenfolge wird die Architektur-Veränderung durch eine simulative oder tatsächliche Implementierung der Maßnahmen schrittweise untersucht. Dabei werden der Grad der Zielerreichung, die Einhaltung der Rahmenbedingungen und der Implementierungsaufwand ermittelt. Daraufhin können eine Bewertung der Lösungsansätze und eine rationale Entscheidung erfolgen.

Für die Restrukturierung des Datenbankzugriffs im Beispiel werden zwei Lösungsansätze entwickelt (siehe Abbildung 4). Der erste Ansatz ist die Ergänzung des ‚Database-Abstraction-Layer' (siehe Abbildung 3) um spezielle Bibliotheken, die angepasste Funktionen für den Zugriff auf DB2 oder Oracle beinhalten. Die existierenden Funktionen des Layers zur Datenmanipulation werden mit den Funktionen der Bibliotheken verknüpft, um einen Zugriff die verschiedenen Datenbank-Systeme zu gewährleisten. Der andere Lösungsansatz sieht die Ergänzung des ‚Database-Abstraction-Layers' durch die Komponente ADOdb [FR05] vor, die Funktionen zur Transformation von SQL-Statements bietet. Die generierten SQL-Statements werden durch ADOdb transformiert und so an die entsprechenden Datenbanken angepasst. Bei beiden Lösungsansätzen stehen die ursprünglichen Funktionen des Layers unverändert zur Verfügung.

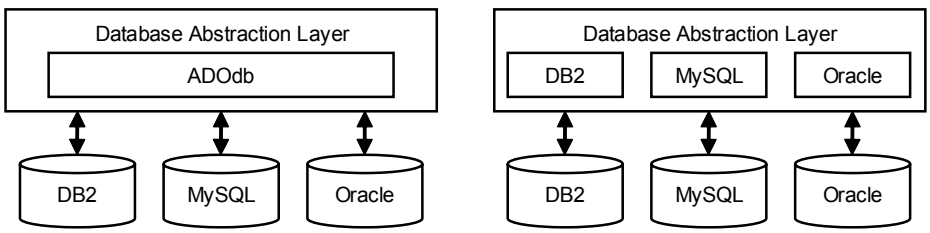


Abbildung 4: Gegenüberstellung der alternativen Lösungsansätze

Die Lösungsansätze unterscheiden sich im Hinblick auf den Implementierungsaufwand und der Antwortzeiten. Während ADOdb als Softwareprodukt direkt integriert werden kann, sind die Bibliotheken mit entsprechenden Zugriffsmethoden erst noch zu entwickeln. Andererseits führen die mit der ADOdb-Alternative verbundenen Transformationen zu einer spürbaren Erhöhung der Zugriffszeiten. Szenarioanalysen des prototypischen Architekturmodells zeigen, dass bei jedem Datenbankzugriff zwei Transformationen durchführt werden; bei einer Datenabfrage wird das SELECT-SQL-Statement *und* der Rückgabewert transformiert, um z.B. den Timestamp an das Oracle-Format anzupassen. Da Antwortzeiten ein kritischer Erfolgsfaktor für Typo3 sind, wird der erste Lösungsansatz der Entwicklung von datenbankspezifischen Bibliotheken besser bewertet und weiter verfolgt. Die weitere Entwicklung von Typo3 als Open-Source-Projekt wird von dieser projektbezogenen Entscheidung jedoch nicht berührt.

## 4 Reduzierung des Analyse- und Bewertungsaufwands

Das hier vorgestellte Vorgehen zur Entscheidungsfindung stellt in Bezug auf den Softwareentwicklungsprozess zusätzlichen Aufwand bei Architekturentwicklung dar, der eine Termin- und Budgeteinhaltung erschwert. Die Aktivitäten Modellerstellung, Ermittlung der Abhängigkeitsbeziehungen, szenariobasierte Bewertung von Lösungsalternativen und schrittweise Untersuchung von Veränderungen müssen deshalb jeweils in ihrem Aufwand minimal gestaltet werden. Während der Entscheidungsfindung muss dazu identifiziert werden, welche Elemente und Beziehungen direkt von der erforderlichen

Architekturveränderung betroffen sind und welche große Auswirkungen auf die restlichen Teile der Architektur haben. Nur diese Teile werden dann beschrieben, untersucht und bewertet. Die Entscheidung, was im Einzelfall relevant ist oder was weggelassen werden kann, erfordert einen Kompromiss, der Einfluss auf die Qualität der Entscheidung und auf die Wahrscheinlichkeit von Fehlentscheidungen hat. Je höher das Risiko einer Entscheidung ist, desto höherer Aufwand kann und sollte in eine systematische Entscheidungsfindung investiert werden.

Im Fall der Typo3-Architekturveränderung wurde zur Reduzierung des Analyse- und Bewertungsaufwands die Untersuchung auf die Abhängigkeitsbeziehungen zwischen dem ‚Typo3-Core' und des ‚Database-Abstraction-Layers' beschränkt. Die Beziehungen zwischen den ‚Extensions' und der Datenbank wurden nicht betrachtet, da die Programmierrichtlinien die Nutzung des Layers vorschreiben. Um Seiteneffekte aufgrund einer Verletzung dieser Richtlinie auszuschließen, wird jedoch vor Beginn der Restrukturierung deren Einhaltung überprüft. Eine grobe Abschätzung des Aufwands gegenüber dem Risiko im Anwendungsumfeld ließ sowohl den Zeit- als auch den Kostenaufwand gerechtfertigt erscheinen.

## 6 Resümee

Durch die konsistente Berücksichtigung der Abhängigkeitsbeziehungen wird es möglich, Architekturentscheidungen systematisch und nach rationalen Gesichtspunkten zu treffen, um Fehlentscheidungen und deren negativen Konsequenzen zu vermeiden. Die Fokussierung auf häufig verwendet Beziehungstypen stellt sicher, dass der Entscheidungsprozess in komplexen Großprojektumgebungen sinnvoll eingesetzt werden kann. Trotzdem sollte bei der Durchführung der Entscheidungsfindung jederzeit das Aufwand-Nutzen Verhältnis berücksichtigt werden.

## Literaturverzeichnis


[BL04] Bengtsson, P.; Lassing, N.; Bosch, J.; van Vliet, H.: Architecture-level modifiability analysis (ALMA), Journal of Systems and Software - Volume 69 - Nr 1-2 - Januar 2004, Seiten 129 - 147.

[EW03] Eisenführ, F.; Weber, M.: Rationales Entscheiden, Springer, Berlin u.a., 2003.

[FR05] Fritz, R., Hinderink D. et al: Typo3: Enterprise Content Management, Packt, 2005.

[KK00] Kazman, R.; Klein, M.; Clements, P.: ATAM - Method for Architecture Evaluation, URL: http://www.sei.cmu.edu/pub/documents/00.reports/pdf/00tr004.pdf, Abruf 2008-10-01.

[KL05] Kruchten, P.; Lago, P.: Building Up and Reasoning About Architectural Knowledge. In: 2nd International Conference on the Quality of Software Architectures (QoSA 2006)

[SA01] Shereshevsky, M.; Ammari, H. et al: Information Theoretic Metrics for Software Architectures, In: '25th Annual International Computer Software and Applications Conference', IEEE Computer Society Los Alamitos (CA, USA), 2001, Seiten: 151 - 160.

[SL99] Simon, F.; Löffler,S. et al: Metrics Based Refactoring, In: 'Proceedings of the Fifth European Conference on Software Maintenance and Reengineering', IEEE Computer Society, Los Alamitos (CA, USA), 1999, Seiten: 30 - 38.