IT-Umgebungen mit IT-Architektur gestalten

Roadmaps herleiten und neue Technologien adäquat einsetzen

Hans-Peter Hoidn[1]

Abstract

Methoden der IT-Architektur sind zwar wohl bekannt, werden jedoch zur Gestaltung von IT-Systemen noch wenig eingesetzt; oft wird auch eine IT-Strategie ohne jeden Bezug zur Architektur der IT-Systeme und der IT-Infrastruktur definiert. Dabei könnte die IT-Architektur wertvolle Beiträge bei der Gestaltung einer IT-Umgebung liefern und mithelfen, neue Vorhaben zu definieren, entsprechende Projekte zu positionieren und eine adäquate Roadmap zu bestimmen. Dieser Artikel vermittelt einen Ansatz für den Einsatz von IT-Architektur zur Gestaltung von IT-Umgebungen. Dabei werden wenige geeignete Methodenansätze ausgewählt und insbesondere darauf geachtet, dass sie die Komplexität verringern und mit geeigneten Visualisierungen die Kommunikation fördern. Ein zentrales Anliegen ist, Business und IT einander näher zu bringen und hierzu praxiserprobte Arbeitsweisen zu vermitteln. Damit verbunden ist die Hoffnung, dass nicht nur die Architekten selbst Architektur gut finden, sondern dass auch Business und Management IT-Architektur zu schätzen wissen.

Architektur wozu?

Dieser Artikel beschreibt erprobte methodische Ansätze um mit Hilfe von IT-Architektur eine Evolution einer IT-Umgebung im heutigen Business- und Technologie-Umfeld zu gestalten. Dabei sind einerseits sich rasch verändernde Geschäftsanforderungen zu berücksichtigen und andererseits neue Technologien einzusetzen. Eine der grössten Herausforderungen ist, die neuen technologischen Trends auf meist inflexible bestehende IT-Umgebungen anzuwenden:

- Durch das Aufkommen mobiler Geräte erwarten Benutzer, dass mobile Apps nun Informationen aus geschlossenen „Silo"-Anwendungen nutzen.
- Mobile Anwendungen und insbesondere das Internet of Things (IoT) verlangen schnelle Antwortzeiten bei gleichzeitiger Explosion der Datenmengen.
- Das Business erwartet, dass die IT mehr Funktionalität zu geringeren Kosten bereit stellt. In Cloud-Technologien werden grosse Hoffnungen gesetzt.

Ein IT-Architektur-Ansatz unterstützt die Arbeit mit den anstehenden Herausforderungen, insbesondre folgende Aspekte:

- IT-Systeme bestehen aus Komponenten, wobei die Funktionalität jeder Komponente entsprechend eingegrenzt wird und Interaktionen zwischen Komponenten wohl definiert sind. Die Komponenten sind somit die Bausteine eines Systems; Komponenten können wiederum rekursiv aus feineren Komponenten bestehen. Die Zuordnung von Funktionalität zu Komponenten klärt den Stand der Dinge und hilft so bei der Weiterentwicklung des Gesamtsystems.
- SOA (Service-Oriented Architecture) ist ein Architektur-Stil, der Geschäftsfunktionen service-orientiert verfügbar macht. Dabei repräsentiert ein Service eine prinzipiell mehrfach anwendbare Geschäftsfunktion; die Geschäftsanforderungen werden so in einem Portfolio von Services konkretisiert. Diese Architektur-Sicht abstrahiert von Ausprägungen wie Microservices und von der Implementierung. BPM (Business Process Management) unterstützt den Ablauf von Geschäftsfunktionen in einem Prozess.

Für die Transformation einer bestehenden IT-Umgebung unter Einbezug neuer Technologien können Methoden der IT-Architektur eine veränderte Sicht herbeiführen. Dies unterstützt die Evolution der IT-Umgebung und trägt schliesslich dazu bei, eine adäquate Roadmap zu definieren. Die vorgestellten Ansätze wurden in verschiedenen Kundensituationen erprobt und verfeinert. Oft genügten zwei- bis dreitägige Workshops, um konkrete Geschäftssituationen zu analysieren und Möglichkeiten einer Weiterentwicklung der IT-Umgebung unter Einbezug neuer Technologien aufzuzeigen.

Beispiel: Als Illustration wird ein *Car Leasing* Prozess verwendet. Der bestehende Prozess existiert nur als eine Abfolge von Arbeitsanweisungen mit dem Nachteil, dass der Ablauf zu keinem Zeitpunkt überblickt werden kann und damit auch der Stand der Arbeiten grundsätzlich nicht sichtbar ist. Die Bank verliert Marktanteile, da der Ablauf bis zu zwei Wochen dauert und Kunden nicht gerne so lange warten.

Architektur-Methoden anwenden

Architektur-Methoden behandeln eine IT-Umgebung als ein System von Komponenten. Neue Technologien werden durch neue Komponenten repräsentiert, die dann in ein neues Gefüge einzubinden sind. Insbesondere für verteilte Systeme stellt eine IT-Architektur mit der Komponenten-Perspektive eine entscheidende Hilfe dar.

Architektur-Methoden werden bereits seit Jahrzehnten erfolgreich eingesetzt. Mit TOGAF (The Open Group Architecture Foundation) [TOG09] existiert ein weltweit akzeptierter Architektur-Standard, der neben einem Framework auch eine entsprechende Terminologie vermittelt. IBMs "Architectural Thinking" betont

Arbeitsresultate und fokussiert auf Dokumente, die eine IT-Architektur definieren und visualisieren; diese Methoden sind von Eeles / Cripps [Eel09] detailliert beschrieben. Brown [Bro16] beschreibt Architektur-Methoden, die insbesondere die Visualisierung betonen.

Sowohl Eeles / Cripps als auch Brown [Bro16] stellen die IT-Architektur eines (grossen) Systems in den Vordergrund und damit ist deren Architektur-Methodik als Ganzes sehr umfangreich. Für unser Vorhaben, eine IT-Umgebung zu gestalten und die Transformation in eine neue Ziel-Umgebung zu unterstützen, wählen wir deshalb gezielt wenige Methoden-Ansätze aus.

Standards berücksichtigen

Ohne Standards ist ein verteiltes System der heutigen Zeit undenkbar. Schon deshalb werden Standards in den letzten Jahren immer wichtiger, sie bilden ein gemeinsames Verständnis mit einheitlichen Begriffen und liefern technologische Grundlagen.

Für die Architektur ist heute TOGAF die anerkannte Referenz und eine gute Vorgabe (insbesondere auch für die Terminologie, für die es auch eine Deutsche Übersetzung gibt). Die dazu gehörende SOA Reference Architecture [TOG11] liefert eine Ausprägung für den Einsatz von SOA vor, womit auch hierfür eine Grundlage für ein gemeinsames Verständnis gegeben ist.

Standards der OMG (Object Management Group) sind ebenfalls von Bedeutung, insbesondre BPMN 2.0 (Business Process Model and Notation) [OMG13] beinhaltet sowohl eine Prozess-Notation als auch die Ausführungssprache für Business-Prozesse.

Business-Sicht einbringen

Obwohl immer wieder bemerkt wird, dass der wesentliche Zweck der IT die Erfüllung von Geschäftsanforderungen ist, muss festgestellt werden, dass ein gemeinsames Verständnis dieser Anforderungen oft nicht geklärt ist. Verschiedene Personen urteilen oft sehr unterschiedlich.

Für die IT-Architektur ist sehr wichtig, dass die Business-Sicht hinreichend geklärt wird und eine Aussprache über die kritischen Punkte stattfindet. Hierzu ist es hilfreich, aus der Business-Sicht sogenannte "Capabilities" zu diskutieren. Gemäss TOGAF werden Capabilities in der Phase „Architecture Vision" (siehe hierzu [TOG09]) behandelt, wobei Vertreter des Business dann qualifizieren, wie gut eine Capability die Geschäftsanforderungen erfüllt. Daraus leitet der Architekt anschliessend einen entsprechenden Handlungsbedarf ab. Beim Business Case muss der Nutzen in den Vordergrund gerückt werden, denn oft sind Kostenersparnisse marginal gegenüber möglichen Geschäftserfolgen, welche um Faktoren grösser sein können.

Beispiel: Wenn der Business-Prozess von *Car Leasing* von zwei Wochen auf einige wenige Tage verkürzt wird und damit die Bank wieder ins Geschäft kommt, dann zählt im wesentlichen das wieder gewonnene Geschäft und rückt die Kosten für die Verkürzung des Prozesses (z.B. durch den Einsatz von BPM) in den Hintergrund.

Zeithorizont verkürzen

Es ist nicht einfach, sich auf das Wesentlichste zu konzentrieren (siehe hierzu z.B. "Psychologie Heute" [Psy15]) und es besteht immer wieder die Tendenz, ein Vorhaben mit Anforderungen zu überladen und damit zu gefährden. Dies gilt auch, obwohl immer wieder betont wird, dass inkrementelles Vorgehen den grossen Würfen vorzuziehen ist.

Erfahrungsgemäss kann hier eine einfache Regel helfen: innerhalb eines kurzen Zeitraums, z.B. nach 9 Monaten muss ein Resultat mit Nutzen für das Business vorliegen. Dieser Zeitraum hat den Vorteil, dass er innerhalb eines Jahres und damit auch in einem Budget-Zyklus geplant werden kann. Dies verhindert, dass hoffnungsvolle Ansätze durch die Ungeduld des Managements gekippt werden. Dies ist insbesondere bei Transformationen mit neuen Technologien von Bedeutung, da das Management bald einmal sehen möchte, ob der nun eingeschlagene Weg erfolgreich ist oder nicht.

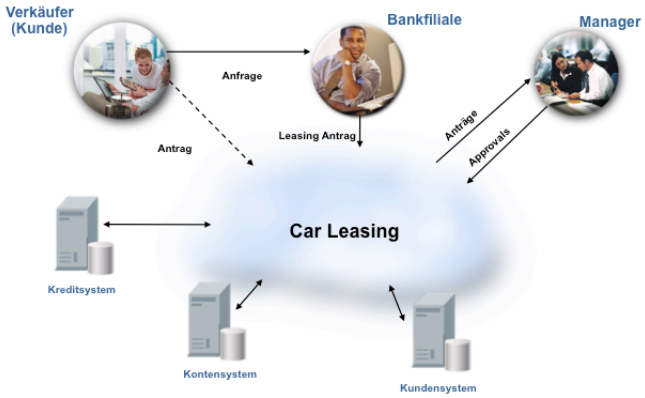
Umfang definieren

Jede Gestaltung von IT-Systemen und insbesondere eine Umgestaltung mit Einbezug neuer Technologien muss verschiedenste Randbedingungen berücksichtigen; dies sind zunächst die bereits existierenden Systeme, die bestehenden vertraglichen Vereinbarungen (u.a. Lizenzverträge), aber auch Restriktionen bezüglich vorhandener Ressourcen. Es ist eine der Aufgaben der Architektur, diese Grenzen transparent zu machen und den Umfang des Vorhabens für einen vorgegebenen Zeithorizont zu definieren. Dabei ist es zentral, dass eine Übereinstimmung darüber besteht, was innerhalb und was ausserhalb des Vorhabens bzw. der zu diskutierenden Lösung ist. Fehlt die Definition des Umfangs dann führt dies erfahrungsgemäss später zu grossen Diskussionen, da verschiedene Personen unterschiedlicher Meinung darüber sein können, was die Lösung beinhalten soll.

Der Umfang kann mit einem sogenannten Kontext-Diagramm illustriert werden; es zeigt das zu behandelnde System als ein geschlossenes Objekt und identifiziert die Schnittstellen zwischen diesem Objekt und externen Systemen bzw. externen Entitäten, wozu auch die agierenden Personen gehören (siehe hierzu [Eel09]). Die Visualisierung des Kontextes klärt das Verständnis der zu lösenden Aufgabe und zeigt die Umgebung auf, worin das anstehende Vorhaben einzubetten ist. Ausserdem hilft das Kontext-Diagramm die Gesamtsicht zu beurteilen und daraus funktionale und insbesondere auch nicht-funktionale Anforderungen der Lösung und u.U. wegen der Schnittstellen auch bezüglich der Systeme ausserhalb abzuleiten (siehe hierzu auch [Bro16]).

Der Vorteil eines Kontext-Diagramms ist, dass wesentliche Einfluss-Parameter für die Schnittstellen diskutiert, bestimmt und festgelegt werden können. Auch werden rasch Konsequenzen deutlich, wird z.B. eine Komponente in den Umfang der anzustrebenden Lösung hinein verschoben, dann kann dies dramatische Folgen haben.

Beispiel: Die Lösung *Car Leasing* implementiert einen Business-Prozess, welcher Daten bestehender Systeme nutzt und auch Daten in bestehende Systeme abspeichert (wie Konten- und Kundensystem). Die bestehenden Systeme haben entsprechende Schnittstellen, die später (u.U. mittels Adapter) implementiert werden.



Figur 1: Beispiel Kontext-Diagramm

Würde z.B. das Kreditsystem in den Umfang der Lösung verschoben, müssten sämtliche Schnittstellen zu allen Komponenten, welche dieses System ansprechen, aufgezeigt werden. Dadurch würde die Anzahl der Schnittstellen explodieren und die Aussicht, eine schlanke Lösung für das Car Leasing zu erhalten, dramatisch sinken.

Entscheidungen transparent machen

Entscheidungen sollten transparent sein und die Grundlagen eines Entscheides entsprechend dokumentiert werden. Wichtige Entscheide sind dann in einem Dokument bzw. an einem Ort aufgeführt und die Grundlagen des Entscheids einschliesslich Motivation und Implikation sowie etwaiger verworfener Varianten sind sichtbar.

Die Dokumentation eines Entscheids findet in einer Tabelle Platz, wie das folgende Beispiel zeigt (welche von der IBMs „Architectural Thinking“ abgeleitet ist – siehe hierzu auch [Eel09]):

Attribute	Value
Thema	BPM (Business Process Management)
Architekturentscheid	Verwendung einer BPM Engine für das Management und die Kontrolle der Business-Prozesse
Anliegen	Business-Prozesse sind zu langsam und niemand hat einen Überblick über den stand eines einzelnen Prozesses
Annahmen	Auf dem Standard BPMN basierende Tools sind verfügbar
Motivation	Die Verwendung von Tools beschleunigt die Modellierung und Implementierung von Business-Prozessen sehr. Dadurch entsteht unmittelbarer Nutzen und die Geschäftssituation kann wesentlich verbessert werden. Die Kontrolle der Prozesse erlaubt eine Optimierung des gesamten Ablaufs.
Alternativen	Kein Support für Business-Prozesse erlaubt keine Verbesserung der Situation
Implikationen	Evaluieren und Lizensieren eines BPM Tools, Kosten

Figur 2: Beispiel Architekturentscheid

Ein Vorteil der Dokumentation der Entscheidungen ist, dass bei Wiedererwägungen die Grundlagen früherer Entscheide eingesehen werden können und so Doppelspurigkeiten vermieden werden. Zimmermann et al [Zim13] geben Anleitungen für gute Entscheidungen, wobei Ausführung-, Konzeption-, Technologie- und Hersteller-Entscheidungen unterschieden werden.

Ist und Soll-Übersichten darstellen

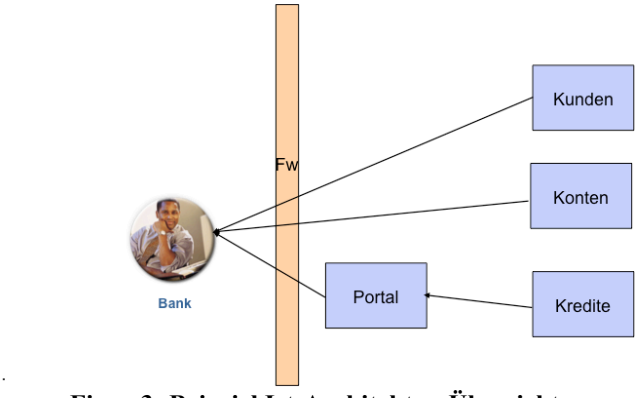
Um ein Vorhaben bestimmen zu können, muss die Ausgangssituation und das Ziel hinreichend genau bestimmt werden; hierzu trägt die IT-Architektur mit entsprechenden Ist- und Soll-Architektur-Übersichten bei (siehe hierzu [Eel09] und [Bro16] unter „Architecture Overview“).

Eine Architektur-Übersicht ist eine Visualisierung der wesentlichen konzeptionellen Komponenten einer Lösung sowie ihrer Verbindungen auf einem passenden Abstraktionsniveau. Dieses muss so gewählt sein, dass verschiedene Personen das Diagramm verstehen. Deshalb ist eine Architektur-Übersicht in der Regel ein Diagramm in einer freien Notation und z.B. nicht ein UML-Diagramm (welches bereits zu viel Detailinformation enthalten würde). Eine gute Architektur-Übersicht vermittelt Sponsoren und externen Betroffenen eines Vorhabens ein konzeptionelles Verständnis sowohl der bestehenden Situation als auch der angestrebten Lösung. Dabei können Varianten diskutiert und mögliche Implikationen abgeschätzt werden, ausserdem wird sichtbar gemacht, wie funktionalen und nicht-funktionalen Anforderungen entsprochen wird. Auch unterstützt eine Architektur-Übersicht die Risiko-Betrachtung.

Erfahrungsgemäss ist es sehr hilfreich, wenn Ist und Soll so dargestellt werden, dass Veränderungen sichtbar und Ergänzungen hervorgehoben sind. Dies fördert eine gute Kommunikation da dies allen Betroffenen die Auswirkungen eines Vorhabens verdeutlicht. Aus einer praktischen Sicht sollte eine Architektur-Übersicht lesbar auf einem Blatt Platz finden. Falls nötig kann eine Hierarchie von Komponenten-Abstraktionen helfen, wobei die hier vorgestellte Architektur-Übersicht die oberste

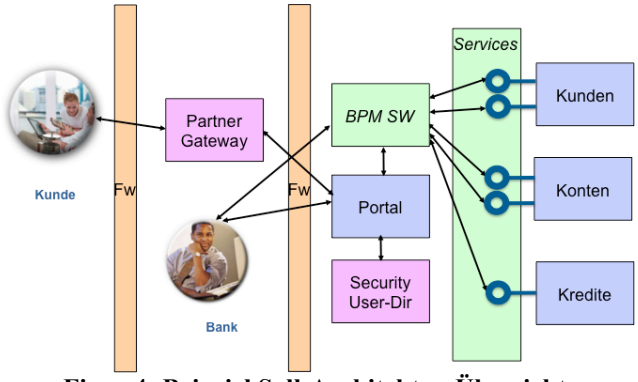
Ebene repräsentiert (siehe hierzu Brown [Bro16], wo eine "Container"- und eine "Component"-Sicht differenziert wird, die Architektur-Übersicht entspricht dann der „Container"-Sicht)

Beispiel: Die Ist-Architektur-Übersicht der bestehenden *Car Leasing* Ist-Situation kann wie folgt dargestellt werden:



Figur 3: Beispiel Ist-Architektur-Übersicht

Die neue Lösung für *Car Leasing* nutzt die bestehenden Anwendungen über Services und benötigt zusätzlich neue Komponenten (u.a. BPM-Software) für das Management der Business-Prozesse. Die Soll-Architektur-Übersicht kann wie folgt dargestellt werden:



Figur 4: Beispiel Soll-Architektur-Übersicht

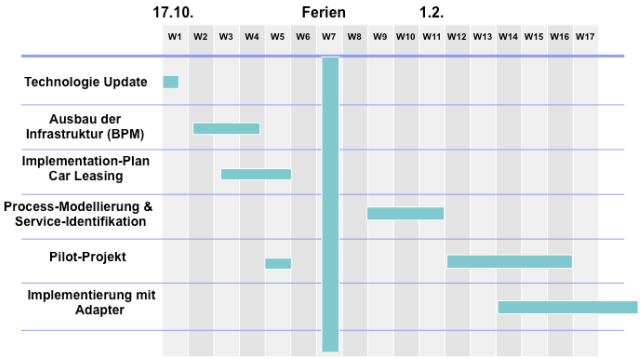
Dieses Diagramm illustriert die Anbindung der bestehenden Systeme und zeigt die Umsetzung mit SOA und BPM auf. Dabei ist verdeutlicht, dass die bestehenden Systeme unverändert über Services angebunden werden. Ausserdem zeigt das Diagramm auf, dass der Kunde nun Teil der Lösung ist, weil er mit dem Ausfüllen eines Formulars in den Business-Prozess eingebunden wird. Damit wird auch das veränderte Geschäftsmodell der Bank illustriert. Hierzu sind allerdings zusätzliche Security-Komponenten notwendig.

Roadmap bestimmen

Eine Roadmap bezeichnet die Aktivitäten, welche die Lücken zwischen einer bestehenden Ist-Situation und einem definierten zukünftigen Soll-Zustand schliessen. Eine solche Roadmap umfasst Projekte sowie weitere notwendige Aktionen (wie Beschaffung von Mittel, Rekrutierung, etc.); notwendige Abklärungen werden ebenfalls in eine Roadmap aufgenommen. Projekte können sich auf Software- und auch Infrastruktur-Vorhaben beziehen.

Erfahrungsgemäss ist es hilfreich, die Differenzen zwischen der bestehenden Situation und dem anzustrebenden Stand als Befund und Empfehlung zu dokumentieren. Dabei werden verschiedene Aspekte der gegenwärtigen Situation analysiert und entsprechende Empfehlungen formuliert. So ist es möglich, verschiedene Anliegen („Concerns") separat zu behandeln.

Eine Visualisierung kann wie folgt aussehen:



Figur 5: Beispiel Roadmap

Fazit

Das hier beschrieben Vorgehen wurde in verschiedenen Kunden-Situationen erfolgreich eingesetzt. In zwei- bis dreitägigen Workshops konnte in den meisten Fällen die Business-Sicht soweit hinreichend konsolidiert werden, dass ein Vorhaben mit entsprechender Roadmap definiert werden konnte.

Die wesentlichen Vorzüge dieses Vorgehens sind:

- Die Kommunikation zwischen Business und IT wird intensiviert und dies fördert das gemeinsame Verständnis. Dadurch können Vertreter des Business und des Managements für die Evolution der IT und dem Einsatz neuer Technologien gewonnen werden.
- Die eingesetzten Architektur-Methoden fokussieren auf die wichtigen Aspekte; sie abstrahieren von den Details der Technologie und der Implementierung.
- Die Komponenten-Sicht der Architektur hilft bei der Einbindung neuer Technologien und unterstützt insbesondere die Definition von Services und Business-Prozessen.
- Die gewählten Diagramme illustrieren hinreichend die weitere Evolution einer IT-Umgebung und helfen für die Kommunikation und bei der Klärung offener Punkte. Insbesondere die Gegenüberstellung von Ist und Soll trägt sehr viel zum gegenseitigen Verständnis bei.
- Die Darlegung der Entscheide sorgt für Transparenz und vermeidet Unsicherheiten.

- Schliesslich zeigt sich, dass das Management eine Einschränkung des Zeithorizontes auf weniger als ein Jahr sehr positiv aufnimmt.

Das beschriebene Vorgehen unterstützt die Evolution von IT-Umgebungen effektiv und effizient und ermöglicht die kontinuierliche Anpassung der IT an Geschäfts-anforderungen und Technologie-Trends. Das hier vorgestellte Vorgehen wird deshalb vom Autor auch im Unterricht und an Kursen weiter vermittelt.

Referenzen

[Bro16] Simon Brown, The Art of Visualising Software Architecture, Draft auf leanpub.com, 2016

[Eel09] Peter Eeles & Peter Cripps, The Process of Software Architecting, Addison-Wesley Professional, 2009, ISBN 978-0321357489

[OMG13] BPMN 2.0.1, Business Process Model and Notation, OMG Standard, 2013, OMG Dokument formal/2013-09-02

[Psy15] Den Alltag managen: Wie Sie mit simplen Regeln Ihr Leben vereinfachen, Psychologie Heute. Ausgabe 11/2015 (Nov. 2015), 18-25

[TOG09] TOGAF, The Open Group Architecture Framework, Version 9.1, The Open Group Dokument C091, 2009

[TOG11] SOA Reference Architecture, The Open Group Dokument C119, 2011

[Zim13] Uwe Zdun, Rafael Capilla, Huy Tran, Olaf Zimmermann, Sustainable Architectural Design Decisions, IEEE Software, November/December 2013, 46-53

---

[1] Hans-Peter Hoidn, Dr. sc. math. ETH, (hans-peter@hoidn.ch) ist The Open Group Distinguished Certified IT Architect, war bis zu seiner Pensionierung Executive Architect bei IBM und ist IT-Architekt seit es diese Rolle in der IT gibt. Heute widmet er sich hauptsächlich der Weitergabe seiner Erfahrungen und seiner Architektur-Kenntnisse in Beratungsmandaten sowie in der Ausbildung mit Lehraufträgen und Kursen (Universität Zürich und HSR Rapperswil); wobei die Gestaltung von IT-Umgebungen mit neuen Technologien – insbesondere mit SOA und BPM – im Vordergrund stehen.