Teil I

Projektarchitektur und Kommunikationsschnittstellen

1 Software- und Projektstruktur

1.1 Komplexität von Projektstrukturen

Wir stellen im Rahmen von Projekten hochkomplexe Software her, die maßgeschneidert die Anforderungen erfüllt. Jedenfalls ist das unser Ziel. Die Zeiten, in denen tolle Projekte von einer einzelnen Person gestemmt wurden, sind lange vorbei. Die Komplexität der Anforderungen und der Integration in bestehende Systemlandschaften erfordert angemessene Projektstrukturen. Diese können schnell ähnlich komplex werden wie die zu realisierende Softwarestruktur (Abb. 1.1).

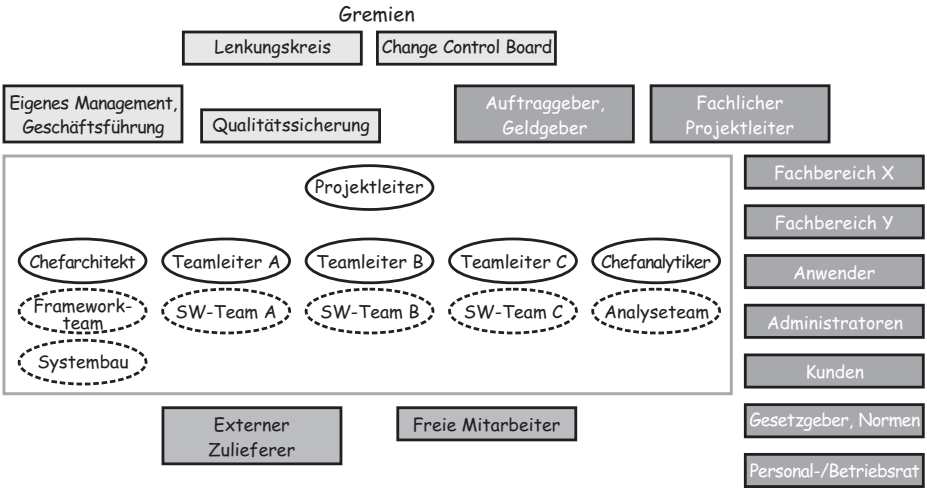

Abbildung 1.1: Heutige Projekte erreichen schnell eine komplexe Projektstruktur.

Wenn wir die Managementsicht aus Abbildung 1.1 technisch weiter auflösen und die Schnittstellen explizit modellieren, erhalten wir eine hochgradig vernetzte Struktur, die wir vermutlich in unserer Software nicht dulden würden (Abb. 1.2). Zu aufwendig wären Wartung und Erweiterungen. Die

abgebildeten Interfaces sind nur Beispiele. Im Einzelfall kann Ihre Realität etwas einfacher oder noch komplexer aussehen.

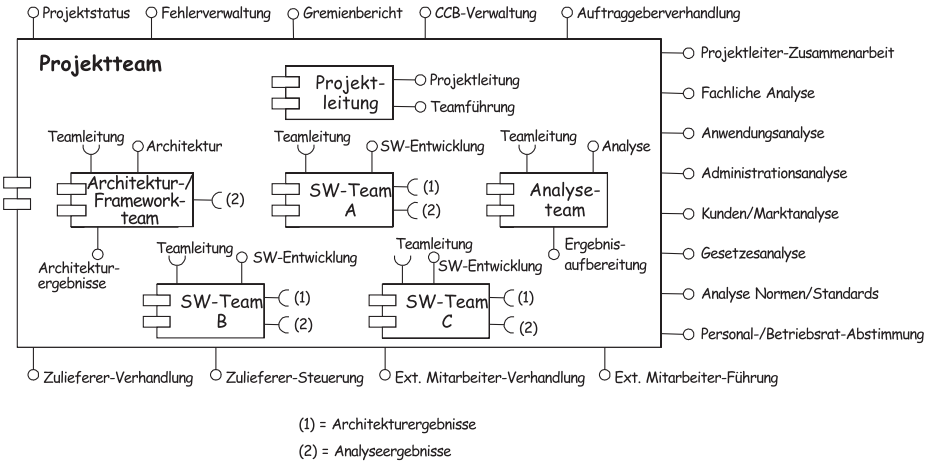

Abbildung 1.2: Die Verfeinerung der Projektstruktur aus Abbildung 1.1 führt zu einem komplexen Kommunikationsnetzwerk. (Beispielhafte Darstellung in UML: Alle Interfaces sind bidirektional.)

Innerhalb unserer Teams finden wir zudem eine Mikrostruktur vor, die eigene Kommunikationsinterfaces ausgebildet hat (Abb. 1.3). Die Gesamtkomplexität wird so noch um eine Stufe erhöht.

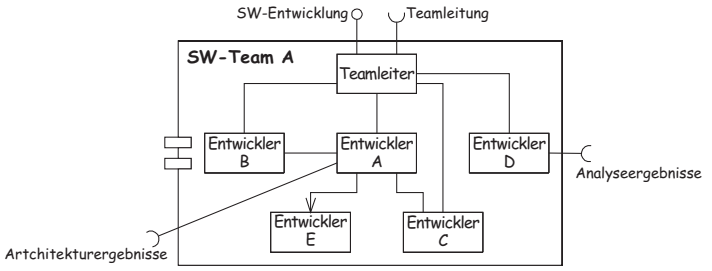

Abbildung 1.3: Zusätzlich zur Struktur aus Abbildung 1.2 finden wir innerhalb unserer Teilteams eine Mikrokommunikationsstruktur vor. (Beispielhafte Darstellung in UML: Alle Interfaces sind bidirektional.)

Glücklicherweise brauchen wir uns der Projektstruktur nicht machtlos zu ergeben. Wir sollten dies auch nicht tun, denn gerade in den Kommunikationsschnittstellen liegt der wesentliche Schlüssel zum Projekterfolg! Genau-

so, wie wir technische Mittel besitzen, um die Abhängigkeiten innerhalb unserer Software in den Griff zu bekommen, gibt es Techniken für die Projektstruktur und Kommunikationsschnittstellen.

Die Themen Selbstorganisation und komplexe Systeme behandeln wir in diesem Buch nicht weiter. Damit haben wir uns eingehend in unserem Buch *Soft Skills für IT-Führungskräfte und Projektleiter* befasst, das wir Ihnen als Weiterführung empfehlen [84].

1.1.1 Was sind Kommunikationsschnittstellen?

Wo liegt nun eigentlich das Problem? Die Schnittstellen sind eindeutig definiert und die erwarteten Ergebnistypen detailliert vorgegeben. Es sollte doch alles klar sein!? Doch egal, wie wir kommunizieren: Der Vorgang lässt sich gut mit einem Filterprozess vergleichen. Bei jedem Transfer bleibt etwas auf der Strecke! Die Anteile können je nach Kommunikationskanal und beteiligten Personen variieren, aber es geht immer Kommunikationsinhalt verloren: Wir haben ein Sender-Empfänger-Problem!

Ein einfaches Kommunikationsmodell beschreibt Kommunikation als Folge von Transformationen [69]. Dabei kann es bei jeder Transformation zu Verlusten im Informationsgehalt kommen. Dies erfolgt sowohl zwischen Personen wie auch innerhalb eines Menschen (Abb. 1.4).

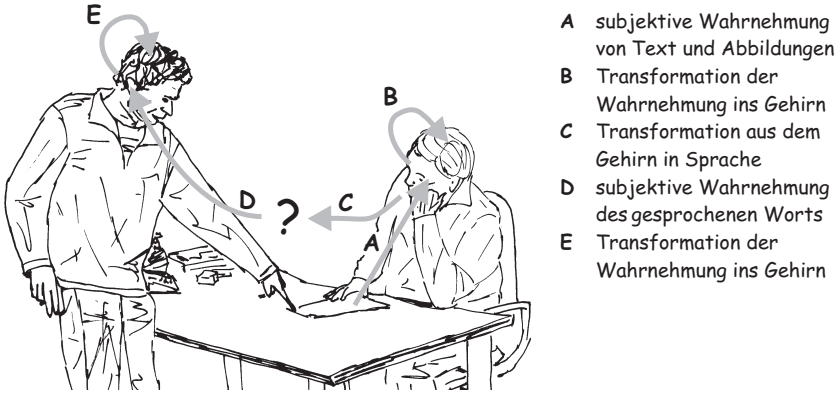

Abbildung 1.4: Kommunikation als Transformations- und Filterprozess: das Kommunikationsmodell nach Shannon und Weaver [69]

Jede Wahrnehmung ist subjektiv. Dazu kommt das Ausfiltern von Informationen aufgrund der physikalischen Einschränkungen unserer Wahrnehmung. Das Wahrgenommene wird transformiert und als Erinnerung im Gehirn abgelegt. Bei dieser Transformation helfen uns unsere bisherigen Erfahrungen. Sie erleichtern uns das schnelle Erfassen, filtern aber erneut

Informationsgehalt aus. Außerdem ist unsere Wahrnehmung durch unsere individuelle Auffassungsgabe begrenzt.

Bei der Rücktransformation aus unserem Gehirn in extern Kommunizierbares, also z. B. Sprache, bleibt erneut einiges auf der Strecke. Besonders bewusst wird uns dies, wenn wir nicht in unserer Muttersprache, sondern in einer Fremdsprache kommunizieren müssen. Wir spüren förmlich, wie Informationsgehalt versickert. Bei unserem Gesprächspartner spielen sich noch einmal dieselben Prozesse ab.

Kommunikationsschnittstellen sind also an sich bereits kritisch. In Teil II ab Seite 41 werden wir Techniken kennenlernen, mit denen wir den Informationsverlust drastisch minimieren können. Das löst zwar noch nicht alle unsere Probleme, doch wir kommen damit deutlich besser voran. Die verbleibenden Probleme beruhen darauf, dass wir es nicht mit technischen Interfaces zu tun haben, sondern mit Menschen.

1.1.2 Andere sind nicht komisch, sondern nur anders!

Wie die Objekte einer Klasse sind auch wir Menschen Individuen. Wir sind nach dem gleichen Bauplan erstellt, aber unsere Attribute lassen einen großen Spielraum an Individualität zu. Kommen wir mit Menschen in Kontakt, so verstehen wir diejenigen schneller, die ähnlich wie wir gestrickt sind. Andere hingegen scheinen uns völlig fremd zu bleiben (Abb. 1.5).

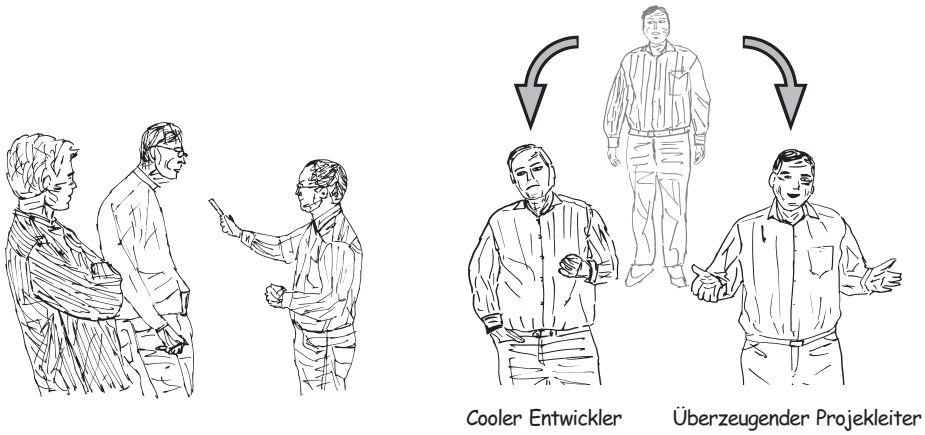

Abbildung 1.5: Menschen sind absolut individuell (links). Derselbe Mensch kann in bestimmten Situationen unterschiedliche Rollen einnehmen (rechts).

In homogenen Gruppen kann dies leicht dazu führen, dass Menschen, die »anders« sind, abgewertet werden. Gruppen von Softwareentwicklern ver-

halten sich da nicht anders. Die Anwender haben sowieso keine Ahnung, die Administratoren machen alles kaputt und der Fachbereich weiß nicht, was er will, obwohl die eigene Lösung doch eigentlich perfekt ist. Und Manager lassen sich von bunten Bildern eher beeindrucken als von detaillierten Informationen. Wie können wir in so einem Umfeld überhaupt arbeiten?

Achtung! Was wir hier vorschnell machen, ist eine abwertende Klassifizierung. Wir packen die Menschen in »Schubladen«, die so negativ vorbelegt sind, dass wir sie dann gar nicht mehr ernst nehmen können. Hier lauert eine enorme Gefahr, die uns isolieren und projektgefährdende Konflikte erzeugen kann!

Schauen wir uns dazu noch einmal Abbildung 1.1 an. Ob wir wollen oder nicht, wir haben es mit einer Reihe von Menschen zu tun, die keinen technischen Background mitbringen. In deren Bereichen sind ganz andere Qualifikationen notwendig. Wenn sie diese nicht hätten, sondern eher technische, wären sie vermutlich unsere Kollegen. Wenn wir ihre Qualifikationen hätten, säßen wir an deren Stelle!

Diese Heterogenität ist besonders wichtig. Um so etwas Komplexes wie ein Softwareprojekt erfolgreich gestalten zu können, sind verschiedenste Qualifikationen erforderlich. Unsere technischen Fähigkeiten sind dafür absolut notwendig, aber eben nicht ausreichend. In Teil III ab Seite 79 wollen wir Ihnen zeigen, wie wir einen angemessenen Zugang zu Menschen aufbauen können, die ganz anders als wir gestrickt sind. Zusätzlich erfahren Sie dabei, wie Konflikte vermieden werden können. Wir werden uns auch »Schubladen« bauen, die im Gegensatz zu unseren bisherigen (Vor-) Urteilen positiv formuliert und belegt sind.

1.2 Bedeutung für IT-Projekte

»Ja natürlich mag es Softwareprojekte geben, in denen die Kommunikation eine besondere Bedeutung hat. Aber doch nicht in meinem Projekt oder meinem Team. Da ist alles klar und geregelt.« Auf solche oder ähnliche Gedanken können Sie nach dem Lesen der ersten Seiten kommen. Und vielleicht ist Ihr Umfeld genau die Ausnahme, die die Regel bestätigt. Aber wie wahrscheinlich ist das?

Vielen Lesern mag es einleuchten, dass die Kommunikation in agilen Projekten einen besonderen Wert hat. Wir gehen jedoch davon aus, dass so gut wie alle Softwareprojekte primär Kommunikationsprojekte sind und sich nur sekundär mit technischen Themen befassen. Verstehen wir wirklich, was die Stakeholder brauchen? Verstehen wir wirklich, welche Probleme der Architekt oder das parallel arbeitende Team sieht? Betrachten wir zur Illustration kurz den Wert von Komunikation und Zusammenarbeit in zwei sehr unterschiedlichen Kontexten.

1.2.1 Agilität: Kleine Projekte, kleine Probleme?

Der Begriff *agil* bedeutet so viel wie *beweglich* oder *leicht zu führen* [20]. Im Agilen Manifest [1] sind die folgenden Prinzipien einer *leichtgewichtigen* Softwareentwicklung festgelegt:

Menschen und Zusammenarbeit vor Prozessen und Werkzeugen
Funktionierende Produkte vor umfassender Dokumentation
Zusammenarbeit mit dem Kunden vor vertraglichen Verhandlungen
Reaktion auf Veränderung vor der Einhaltung eines Plans

Die Punkte auf der rechten Seite der Aussagen sind wertvoll, aber kein Selbstzweck. Sie sollen die wichtigeren Punkte der linken, hervorgehobenen Seite sinnvoll und angemessen unterstützen. Dabei sind die Art und der Grad des Einsatzes der unterstützenden Maßnahmen genau zu bestimmen. Was steckt hinter den vier Punkten? Es gibt für die Toolfrage keine *Silver Bullet*. Wichtig ist, wie wir im Prozess oder mit Werkzeugen arbeiten. Unser methodisches Wissen ist gefragt. Es kann weder durch ein Vorgehensmodell noch durch ein Tool ersetzt werden: *A fool with a tool is still a fool!*

Was nützen vollständige Analyse- und Designdokumente, wenn unsere Software nicht einsatzfähig ist, weil sie entweder nicht das macht, was eigentlich gebraucht wird, oder aber zu fehlerhaft ist? Ein umfangreiches Vertragswerk scheint eher zum Verstecken der wenigen relevanten Abschnitte zu dienen als zur Klarheit. Selbst mit noch so viel Aufwand und den besten Rechtsanwälten wird uns kein wasserdichter Vertrag gelingen. Und wenn doch, was machen wir, wenn unser Softwarepartner in Konkurs geht?

Was nützt es dem Auftraggeber, wenn der Plan eingehalten wurde, aber sich in der Zwischenzeit die Anforderungen geändert haben? Dies kann bei einer falschen Priorisierung von Bewertungsfaktoren zu seltsamen Auswüchsen führen.[1] Bei der Bewertung von Aktiengesellschaften kann es dazu kommen, dass, nur um die geplanten Gewinne eines Quartalsberichts einzuhalten, auf Gewinne verzichtet wird, weil von einigen Börsianern auch positive Abweichungen negativ beurteilt werden [12]. So etwas sollte uns Entwicklern nicht passieren. Dafür gibt es vier zentrale Prinzipien agiler Softwareentwicklung, die helfen können, diese Probleme zu vermeiden [3]:

- Mut
  - Vertraue darauf, Probleme, die morgen auftreten, auch morgen lösen zu können.
  - Spreche aktuelle Probleme noch heute offen und konstruktiv an.

[1] In einer Studie konnten wir 2009 zeigen, das gerade erfolglose Projekte das Ziel der Umsetzung der *ursprünglichen* Anforderungen besonders oft erreichen, wogegen erfolgreiche Projekte die *aktuellen* Anforderungen umgesetzt haben [85].

- Kommunikation
  - Sorge dafür, dass sich die Menschen persönlich kennenlernen.
  - Interveniere bei sozialen Problemen zwischen Beteiligten sofort, denn Kommunikationsprobleme sind dringlich.
- Feedback
  - Vergewissere dich regelmäßig, auf dem richtigen Weg zu sein.
  - Entwickle im Team.
  - Führe Reviews mit der jeweiligen Zielgruppe wie z. B. anderen Teams, Fachexperten, Designern, Anwendern durch.
  - Führe Akzeptanztests durch.
- Einfachheit
  - Suche die einfachste Lösung, um die Anforderungen zu erfüllen. Sie wird immer noch kompliziert genug sein.
  - Entwickle für drei ähnliche Probleme lieber drei verschiedene Lösungen als einmal eine komplexe, generische Universallösung, die doch nicht alle Anforderungen erfüllt und um ein Vielfaches teurer werden wird.
  - Erst beim vierten Mal bist du dir ausreichend sicher, dass du eine abstrakte, generische Lösung erstellen kannst.[2]
  - Vorher ist meist der Aufwand für die Erstellung der generischen Lösung höher als für die drei Einzellösungen.

Die Kommunikation und damit die Kommunikationsfähigkeit wird also als zentraler Erfolgsfaktor gesehen. Dies deckt sich mit unserer Erfahrung aus diversen Negativbeispielen. Obwohl agile Projekte in der Regel von der beteiligten Personenzahl her kleine Projekte sind, spielt die direkte Kommunikation und Kommunikationsfähigkeit bei allen Beteiligten eine überproportional wichtige Rolle.

Schnell kommt es zu Missverständnissen zwischen Entwicklern und Nicht-Entwicklern aus den Fachbereichen oder dem Management. Dadurch entwickeln wir die falsche Software, und die Manager verstehen nicht, was eigentlich vor sich geht. Die Situation wird noch verschärft, indem sowohl die Fachbereiche als auch das Management erheblich mit ihren Erfolgen von uns Entwicklern abhängig sind. Sie sind also abhängig von Menschen, die sie nicht verstehen! Eine solche Situation wird beinahe zwangsläufig zu Ängsten führen – und Angst ist ein schlechter Berater.

[2] Über die Anzahl *vier* lässt sich streiten. Hier geht es weniger um Redundanzfreiheit im Code als um den Sinn und Widersinn generischer Lösungen.

Es kann auch sein, dass unser Gesprächspartner blockiert und wir nicht an die Informationen herankommen, die wir benötigen. Und wir merken das noch nicht einmal. Unser Auftreten kann so weit führen, dass wir uns politische Gegner schaffen, mit denen wir dann Machtspiele austragen, anstatt uns auf unser Projekt zu konzentrieren. Auch innerhalb unseres Entwicklungsteams entgehen uns schnell durch mangelhafte Kommunikation mögliche Synergien. So werden wir kaum eine effiziente Lösung finden können.

Dieses Buch befasst sich im Wesentlichen mit den direkten Kommunikationstechniken. In jedem Teil werden Sie daher Techniken kennenlernen, die Ihnen besonders in agilen Projekten helfen können.

1.2.2 SOA: Großprojekte und direkter Kontakt

Hinter SOA (Service-oriented Architecture) steckt eine Architektursicht zur flexiblen Umsetzung von Geschäftsprozessen in Software. Häufig etwas vorschnell wird SOA auf ein technisches Problem zur Bereitstellung von Webservices reduziert. Schauen wir uns erfolgreiche SOA-Projekte an, stellen wir fest, dass die Technologie gegenüber der Methodik häufig eher sekundär ist [73].

Der Rahmen für SOA-Projekte liegt entgegengesetzt zu den vorher behandelten agilen Projekten. Das Ziel ist dabei die Flexibilisierung der Softwareunterstützung von Geschäftsprozessen in Großfirmen und Konzernen. Waren früher Prozesse über lange Zeiträume konstant, fordert der Markt heute deutlich schnellere Reaktionen, sodass Prozesse durchaus alle paar Monate angepasst werden können (Abb. 1.6).

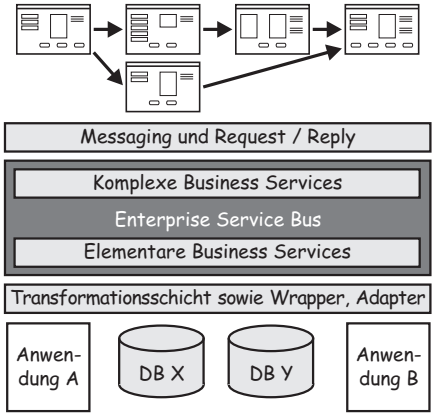

Abbildung 1.6: SOA ist mehr als die Integration von Anwendungen. Durch die Zerlegung in Services kann die Flexibilität der IT-Lösungen erhöht werden, um der Dynamik der Anforderungen besser gerecht zu werden.

Aus technischer Sicht wird mit SOA der bestehende heterogene Zoo von Anwendungen und Informationssystemen durch einen zentralen Enterprise Service Bus gekapselt, der als Schnittstelle für die Anwendungen *Services* zur Verfügung stellt.

Die zentrale Fragestellung lautet: Wie kommen wir zu den einzelnen Services? Dies kann nur in enger Zusammenarbeit mit den Fachbereichen erfolgen, um hinterher auch das angestrebte Ziel der Flexibilität zu erreichen. Ausgangspunkt sind die Geschäftsprozesse, die weiter zerlegt werden und zu statischen wie dynamischen Modellen führen, aus denen dann die konkreten Services abgeleitet werden. Um die gewünschte Dynamik an Änderungen und Erweiterungen realisieren zu können, erfolgt diese methodische Ableitung der Services im direkten Kontakt aller Beteiligten in Form von Workshops. Dabei werden die Geschäftsprozesse analysiert und, falls notwendig, IT-kompatibel modelliert. Des Weiteren werden in einem Domänenmodell Verantwortlichkeiten sowie Schnittstellen festgelegt (Abb. 1.7). Mit den so gefundenen Services können dann (hoffentlich) schnell Prozessanpassungen in unseren Anwendungen erfolgen, indem die Services als Bausteine aufgefasst und neu rekombiniert werden.

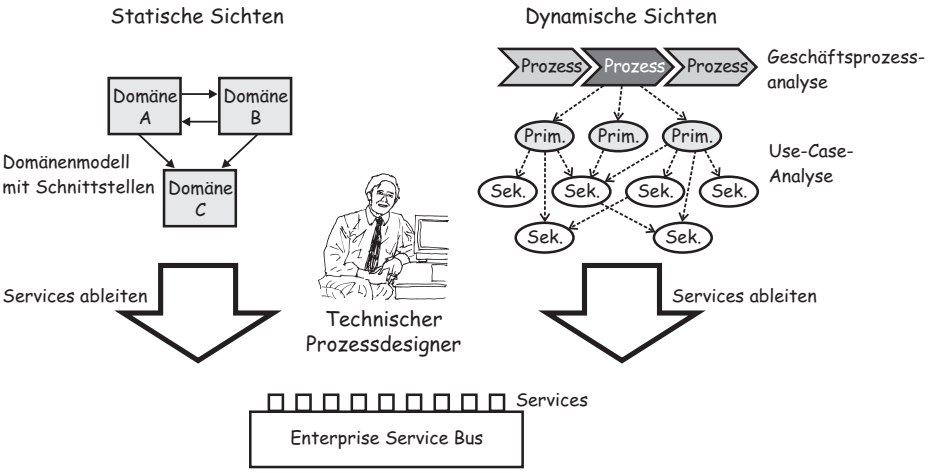

Abbildung 1.7: Die Entwicklung einer SOA ist primär ein organisatorischer und geschäftsprozessorientierter Ablauf.

Dabei wird auf IT-Seite eine neue Rolle definiert, eine Art *technischer Prozessdesigner*. Aufgrund der hohen Dynamik der Anforderungsänderungen ist das primäre Arbeitsmittel des technischen Prozessdesigners die Durchführung von Workshops. Er transformiert die Ergebnisse der Workshops in technische Modelle und stimmt diese mit den fachlichen Ansprech-

partnern ab. Meist wird diese Aufgabe von einem kleinen Kernteam wahrgenommen. Für diese Rolle sind offensichtlich neben technischen und methodischen Fähigkeiten starke kommunikative Fertigkeiten notwendig.

In der direkten Kommunikation der Beteiligten werden die Abläufe optimiert und angepasst. Der Prozessdesigner sorgt für die Transformation der fachlichen Sicht in eine technische und für eine angemessene Modellierung. Damit ist er verantwortlich für die Dokumentation der statischen und dynamischen Sichten auf die Abläufe in den betrachteten Bereichen.