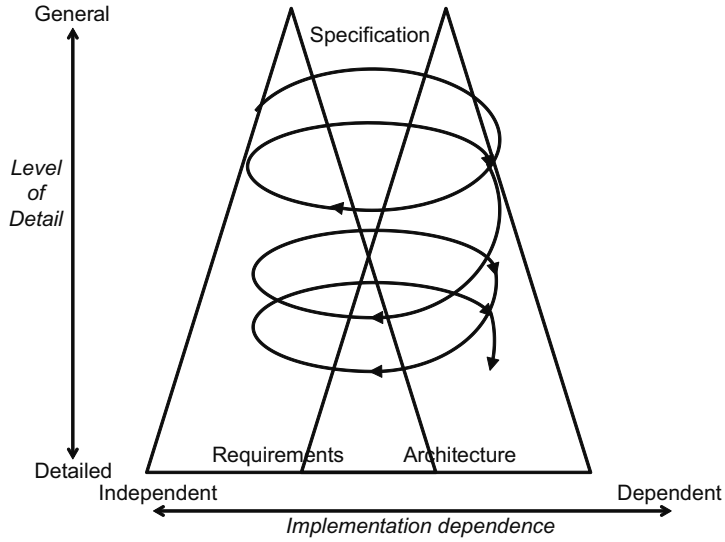

Abb. 2–14 *Das »Twin Peaks Model« [Nus01]*

So können beispielsweise die Auswirkungen von Anforderungen im Hinblick auf den Aufwand erst dann realistisch abgeschätzt werden, wenn eine erste Grobarchitektur zur Verfügung steht. Präsentiert man dann die auf Basis der Grobarchitektur abgeschätzten Aufwände dem Kunden, so ist dieser nicht selten gerne bereit, auf die eine oder andere Anforderung zu verzichten. Umgekehrt können aber mögliche existierende Entwurfsalternativen dem Anforderungsträger aufzeigen, dass bei den Anforderungen noch die eine oder andere Lücke klafft (siehe auch [HM++07]).

2.4.1 Ziele und Aufgaben des Softwarearchitekturentwurfs

Kernaufgabe des Softwarearchitekturentwurfs ist es, einen Konstruktionsweg zu finden, mit dem die funktionalen und nicht funktionalen Anforderungen aus dem Requirements Engineering in einer fertig konstruierten Lösung umgesetzt sind. Dieser Weg ist aber keine Einbahnstraße, wie oben anhand des Twin Peaks Model in Abbildung 2–14 dargestellt.

Vielmehr ist es ein wechselseitiges Aufeinander-Zugehen vom »Wunschkonzert« des Requirements Engineering und dem »Lagerplatz« des Softwarearchitekten mit existierenden Lösungskomponenten, Bausteinen und anderen Architekturartefakten. Wie Abbildung 2–15 illustriert, erarbeitet sich der Softwarearchitekt auf diesem Weg den Bauplan für das zukünftige Softwaresystem in Abstimmung mit allen beteiligten und relevanten Stakeholdern, wie z. B. Requirements Engineer, Kunde, Nutzer, Entwickler, Tester oder Betreiber.

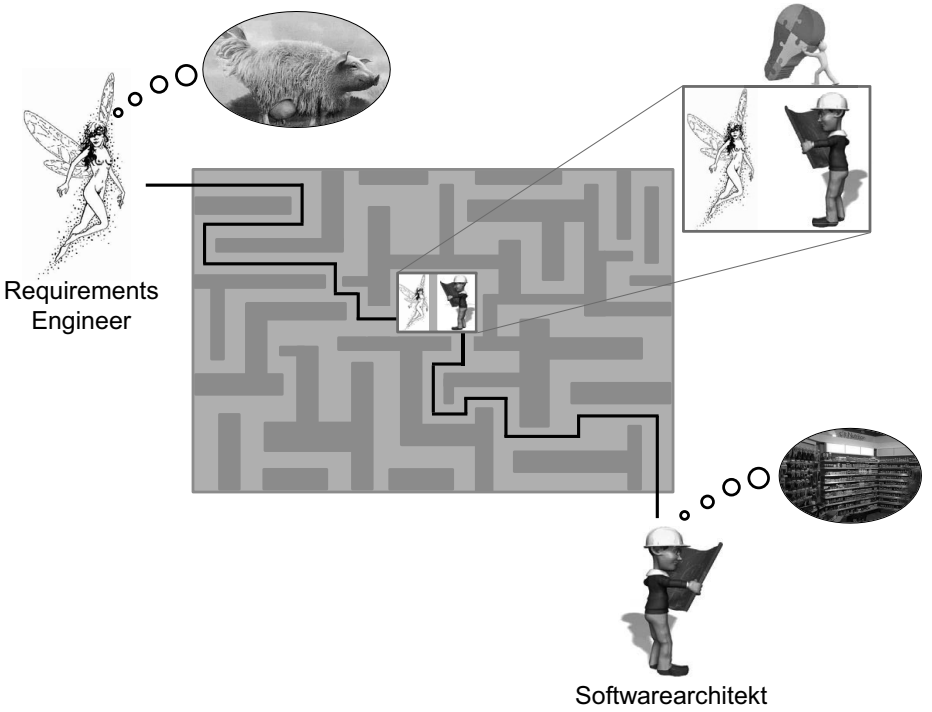

Abb. 2–15 *Der lange Marsch des Softwarearchitekten*

Da ein neu oder weiter zu entwickelndes System nie isoliert zu betrachten ist, müssen dabei die Schnittstellen zu anderen Systemen, den betroffenen Organisationen, den Ausführungsplattformen wie auch der Umsetzungsinfrastruktur berücksichtigt werden. Die Schnittstellen, Vorgaben und Randbedingungen durch die vier angrenzenden Bereiche (siehe Abb. 2–13) sind vom Architekten entsprechend zu berücksichtigen und mitzugestalten.

Darüber hinaus hat der Architekt den gesamten Lebenszyklus des zu entwickelnden Systems wie auch der genannten umgebenden Bereiche mit zu berücksichtigen. Denn Architektur bedeutet immer auch Investition in tragende Elemente, um an anderen Stellen Flexibilität und Erweiterbarkeit zu gewinnen. Dies bedarf aber der Betrachtung der entsprechenden Lebenszyklen, sonst ist die Investition nicht sicher.

2.4.2 Der Softwarearchitekturentwurf im Überblick

Softwarearchitektur wird nicht im stillen Kämmerlein entwickelt, sondern erfordert Teamarbeit, die darüber hinaus mit vielen Projektbeteiligten diskutiert wird. Deshalb ist es notwendig, im Projekt selbst wie auch im Projektumfeld ein gemeinsames Verständnis darüber zu entwickeln, was Bestandteil einer Softwarearchitektur ist und was nicht. Dazu gehört auch eine gemeinsame Nomenklatur für die wichtigsten Begriffe wie z. B. Schnittstelle oder Baustein.

Der Architekt hat dabei nicht nur Schnittstellen mit dem Requirements Engineering zu berücksichtigen, sondern auch noch mit anderen Disziplinen und Rollen in der Softwareentwicklung, wie in Abbildung 2–13 dargestellt. Ausgehend von den Randbedingungen und Vorgaben dieser umgebenden Bereiche entwirft der Softwarearchitekt die Softwarearchitektur. Damit legt er die wesentlichen Aspekte der Lösung fest, wie z. B. Bausteinstruktur und Interaktionsmuster. Dabei ist der Architekturentwurf für sich allein betrachtet bereits kein sequenzieller Prozess. Selbst ein iteratives Vorgehen wird dem Wesen des Architekturentwurfs nicht gerecht. Man kann die einzelnen Aufgaben im Architekturentwurf nicht sinnvoll in eine lineare Reihenfolge bringen. Stattdessen teilen wir, wie Abbildung 2–16 zeigt, den Architekturentwurf in vier gleichberechtigte Tätigkeiten auf:

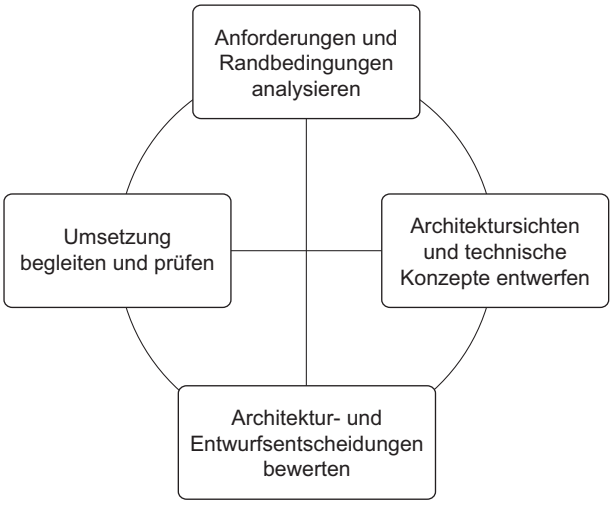

Abb. 2–16 *Iterativ und inkrementell durchgeführte Schritte beim Architekturentwurf*

- **Anforderungen und Randbedingungen analysieren:**
  Zentrale Aufgabe der Architekturanalyse ist es, die Ziele, Randbedingungen sowie die funktionalen und insbesondere die nicht funktionalen Anforderungen aus dem Requirements Engineering im Kontext der anderen umgebenden Bereiche (vgl. Abb. 2–13) zu untersuchen. Dabei müssen Qualität sowie Flexibilität (Stakeholder ist offen gegenüber Änderung) und Änderbarkeit (kann

sich durch externe Einflüsse im Laufe der Zeit ändern) der Anforderungen überprüft werden. Lücken in den Anforderungen müssen aufgedeckt werden [HNS99]. Gerade bei den nicht funktionalen Anforderungen muss hier noch meist nachgebessert werden, da die Anforderungsträger diese häufig als selbstverständlich verstehen. In Zusammenarbeit mit allen Projektbeteiligten – insbesondere den Designern und Entwicklern – ist ein erstes Verständnis für Architekturstil und technische Infrastruktur zu entwickeln. Dies ist die zentrale Architekturmetapher des Systems.

- **Architektursichten und technische Konzepte entwerfen:**
  Hier wird die Architektur detaillierter ausgearbeitet. Es erfolgt insbesondere auch die sichtenbasierte Beschreibung der unterschiedlichen Architekturebenen wie z. B. der fachlichen und der technischen (vgl. auch Abb. 2–13). Dabei gilt es, die funktionalen Anforderungen auf die entsprechende fachliche Architekturebene herunterzubrechen sowie für die relevanten Aspekte aus den nicht funktionalen Anforderungen entsprechende querschnittliche Lösungsbausteine in der technischen Architekturebene zu konzipieren und zu dokumentieren (vgl. auch Kap. 4). Der grundsätzliche Lösungsrahmen aus dem Architekturstil und der technischen Infrastruktur muss dabei entsprechend beachtet werden.
- **Architektur- und Entwurfsentscheidungen bewerten:**
  Die erarbeitete Architektur muss qualitätsgesichert werden. Hier können die unterschiedlichen Techniken angewendet werden, angefangen von den diversen Reviewtechniken über technische Prototypen und Tests bis hin zu analysierenden und bewertenden Verfahren. Entscheidend hierbei ist, dass man aus den Anforderungen konkrete Szenarien abgeleitet hat, um die Qualität der Architektur sicherzustellen (siehe auch Kap. 5).
- **Umsetzung begleiten und prüfen:**
  Von ihrer Bedeutung her häufig unterschätzt ist die vierte Aufgabe des Architekten: das Kommunizieren der Softwarearchitektur an alle Projektbeteiligten. Nur wenn alle – vom Entwickler bis zum Kunden – die Softwarearchitektur verstanden und akzeptiert haben, wird sie erfolgreich umgesetzt werden können und so ihre erhoffte Wirkung entfalten. Dabei muss die Softwarearchitektur natürlich so kommuniziert werden, dass der Kommunikationspartner diese auch verstehen kann. Das heißt, dem Kunden wird man die Architektur auf einem anderen Detaillierungsgrad erklären als dem Entwickler. Auch dies ist keine Einbahnstraße, sondern ein Voneinander-Lernen und Verstehen. Bei Begleitung der Architekturumsetzung wird diese mit den Projektbeteiligten reflektiert und so werden ggf. noch offene Punkte, Verbesserungspotenzial, Fehler und Weiterentwicklungsrichtungen identifiziert.

Auf Basis dieser Konzepte sollte auch eine effektive Werkzeugunterstützung aufgebaut werden. Mit ihr können die einzelnen Tätigkeitsbereiche des Softwarearchitekturentwurfs möglichst optimal unterstützt werden, beispielsweise zur Analyse und Verwaltung der Anforderungen, zur Bearbeitung von Architekturmodellen und der Dokumentation, zur Qualitätssicherung und schließlich zur Kommunikation. Für diese einzelnen Aufgabenbereiche existieren bereits eigenständige Werkzeuglösungen (siehe auch Kap. 6). Diese isolierten Werkzeugunterstützungen müssen möglichst nahtlos integriert werden, damit sie dem Architekten effektiv zur Verfügung stehen. Besonders vor dem Hintergrund, dass der Entwurfsprozess nicht sequenziell, sondern eher iterativ und inkrementell, ja sogar nebenläufig durchgeführt wird, ist es besonders wichtig, die Lücken zwischen den einzelnen Werkzeugen zu schließen.

2.4.3 Wechselspiel der Tätigkeiten und Abstraktionsstufen im Entwurf

Wie bereits dargestellt, können die einzelnen Tätigkeiten im Softwarearchitekturentwurf nicht sinnvoll in eine lineare Ordnung gebracht werden. Vielmehr sind sie gleichberechtigte Bereiche, denen sich der Softwarearchitekt je nach konkreter Projektsituation entsprechend zuwenden muss. Der Architekturentwurf ist ein stetiges Wechselspiel zwischen den Tätigkeiten aus Abbildung 2–16:

- Anforderungen und Randbedingungen analysieren
- Architektursichten und technische Konzepte entwerfen
- Architektur- und Entwurfsentscheidungen bewerten
- Umsetzung begleiten und prüfen

Im Rahmen des Architekturentwurfs führt der Softwarearchitekt diese vier Tätigkeiten quasi gleichzeitig in beliebiger, für ihn sinnvoller Abfolge entsprechend den Projektnotwendigkeiten und dem -kontext aus. Dieses iterative und inkrementelle Wechselspiel der Tätigkeiten geht einher mit einem Top-down- und Bottom-up-Wechsel der Abstraktionsstufen und Perspektiven aus Abbildung 2–13:

- Vorgaben und Randbedingungen der umgebenden Bereiche
- Softwarearchitektur mit Abstraktionsstufen, beispielsweise:
  - Architekturstil und technische Infrastruktur
  - Fachliche und technische Architekturebene
- Programmdesign und Implementierung

So wechselt man im Architekturentwurf von der obersten Abstraktionsstufe der Randbedingungen und Vorgaben der umgebenden Bereiche über die Abstraktionsstufen in der Softwarearchitektur – im Beispiel der Architekturstil und die technische Infrastruktur sowie die fachliche und technische Architekturebene – bis zu der untersten Abstraktionsstufe, dem Softwareprogramm selbst, dem Pro-

grammdesign und der Implementierung. Der Architekturentwurf ist somit ein kontinuierlicher Fluss zwischen top-down und bottom-up innerhalb der Abstraktionsstufen, der einhergeht mit einem stetigen Wechsel der Tätigkeiten, die iterativ und inkrementell durchgeführt werden. Dieser Zusammenhang ist in Abbildung 2–17 illustriert.

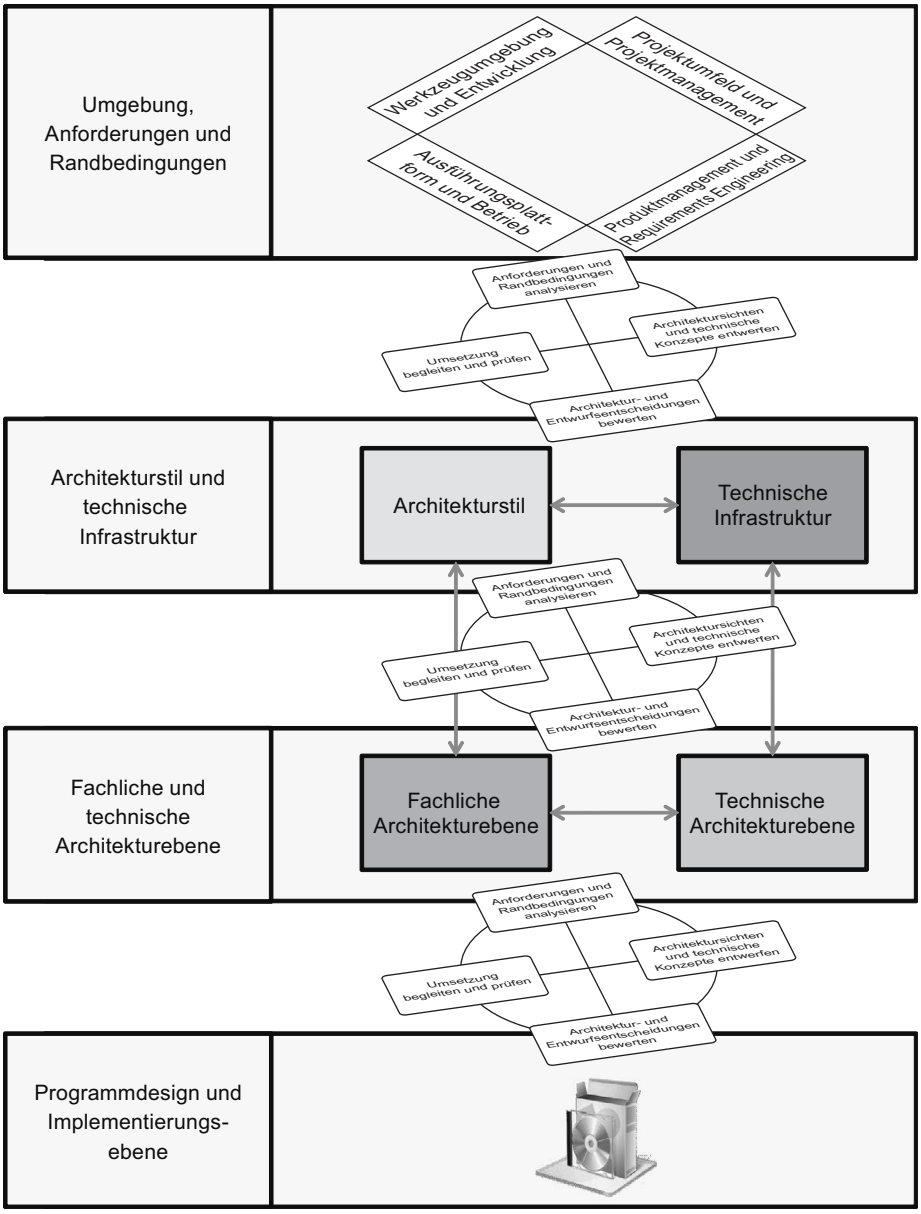

Abb. 2–17 *Der Softwarearchitekturentwurf im Überblick*

Dabei werden vom Projektumfeld und Projektmanagement der Projektrahmen und die Randbedingungen für den Architekturentwurf vorgegeben. Hier liefert der Architekturentwurf etwas zurück, z. B. technische Projektrisiken oder Planungsinformationen.

Destilliert durch das Requirements Engineering kommen vom Anforderungsträger die funktionalen und nicht funktionalen Anforderungen für den Architekturentwurf. Dieses ist aber keine Einbahnstraße, wie bereits oben dargestellt. Vielmehr spiegelt der Architekturentwurf die Möglichkeit der Realisierbarkeit der Anforderungen und deren Konsequenzen für das Requirements Engineering zurück.

Darüber hinaus müssen bereits beim Architekturentwurf die teilweise oder vollständig vorhandene technische Infrastruktur, beispielsweise Serverinfrastruktur, Betriebssysteme oder Vorgaben für Programmiersprachen, und die Betriebsorganisation berücksichtigt werden. Gerade die Schnittstelle zum Betrieb wird bei der Entwicklung viel zu häufig vernachlässigt, was zu großen Problemen beim Ausrollen führen kann. Und wiederum gilt, dass dies keine Einbahnstraße ist: Der Architekturentwurf kann zu neuen Anforderungen an die technische Infrastruktur führen, wie z. B. einem weiteren Ausbau der Serverlandschaft oder der Integration einer weiteren Middleware.

Und schließlich liefert der Architekturentwurf die Vorgaben für das Feindesign und die Programmierung. Aber auch hier sind Rückflüsse möglich und notwendig. So können bei der Umsetzung der Softwarearchitektur unvorhergesehene Probleme auftreten. Werden diese nicht dem Softwarearchitekten kommuniziert, sondern eine vermeintlich einfache Lösung umgesetzt, so kann dies die ganze Architektur unterwandern. Deshalb ist es wichtig, dies mit dem Softwarearchitekten zu diskutieren und gemeinsam eine Lösung zu entwickeln, die unter Umständen auch zu einer Änderung in der Architektur führen kann.

2.4.4 EXKURS: Aufgaben des Softwarearchitekten und Bezug zu anderen Rollen

Aufgabe des Architekten ist es, aus den funktionalen und nicht funktionalen Anforderungen unter Berücksichtigung der Vorgaben und Randbedingungen der anderen umgebenden Bereiche einen Bauplan für das System zu entwickeln. Nach diesem Bauplan richten sich dann Erstellung, Wartung, Pflege und Weiterentwicklung aus. Hierfür ist eine vollständige und prägnante Architekturbeschreibung zu erstellen. Die Architekturbeschreibung dient dabei einerseits als Kommunikations- und Diskussionsplattform, andererseits auch als Design- und Implementierungsplan. Wie Abbildung 2–18 zeigt, hat der Architekt dabei eine Vielzahl von Schnittstellen zu fast allen Rollen in einem Softwareentwicklungsprojekt zu bedienen.

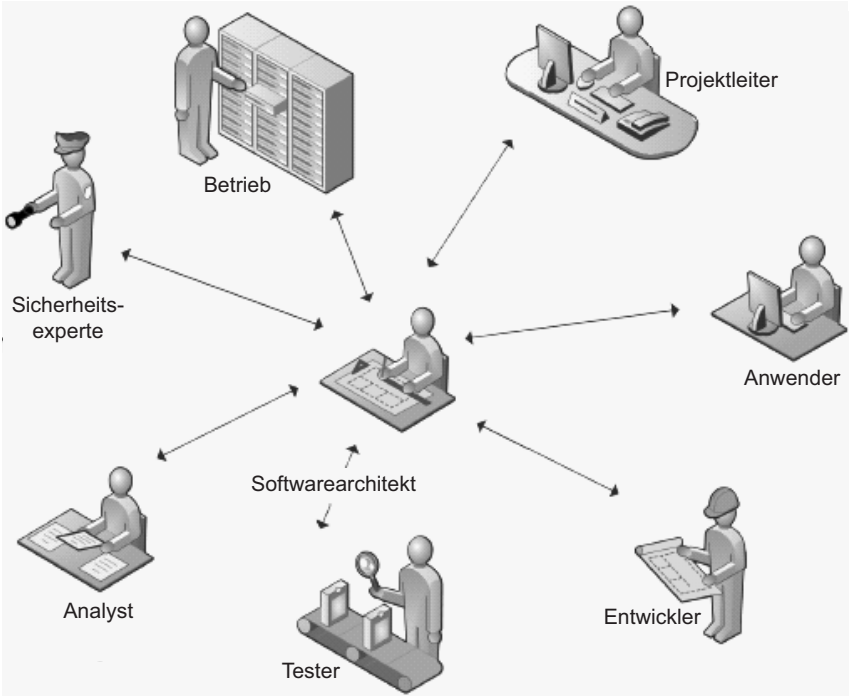

Abb. 2–18 *Der Softwarearchitekt in Bezug zu den angrenzenden Rollen*

- **Kommunikations- und Diskussionsplattform:**
  Anhand der Architektur stellt der Architekt gegenüber dem Requirements Engineer, dem Kunden und ggf. dem Nutzer die Machbarkeit der Anforderungen dar. Dabei unterstützt er bei dem Zuordnen, Priorisieren und Reflektieren von funktionalen und nicht funktionalen Anforderungen. Er kann Widersprüche erkennen und letztlich dafür garantieren und sorgen, dass die Anforderungen umgesetzt werden können.

Er zeigt Integrationsmöglichkeiten von existierenden Lösungen und Systemen auf und gleicht die Anforderungen mit der existierenden Systemarchitektur und Hardware ab. Er erarbeitet, evaluiert und bewertet alternative Lösungsansätze. Und schließlich berät der Architekt anhand der Softwarearchitektur den Projektleiter bei der Projekt- und Iterationsplanung, unterstützt die Risikoanalyse und -vermeidung und hilft so bei der Definition der Arbeitsstrukturierung und -verteilung.

- **Design- und Implementierungsplan:**
  Für die Entwickler ist der Architekt ein zentraler Ansprechpartner. Er legt die Bausteine des Systems sowie deren Schnittstellen und Interaktionsmuster fest. Die Integration von neuen Technologien und innovativen Lösungsansätzen muss er vorantreiben und mit den Entwicklern diskutieren. Er leitet die Erarbeitung, Einführung, Schulung und Überprüfung von Programmierrichtlinien. Er hilft den Entwicklern bei der Erstellung von Prototypen und exemplarischen Teillösungen und forciert die Wiederverwendung von existierenden Implementierungen und Teilimplementierungen. Er erklärt die Architektur, macht Entwicklungsvorgaben, gibt seine Erfahrung weiter und führt Codereviews durch. Ebenso unterstützt er die Tester. Im Idealfall gibt er sogar Testrahmenbedingungen und konkrete Testfälle vor, insbesondere solche zur Überprüfung der konkreten Architekturziele. Er hilft bei der Festlegung von Reihenfolgen und Testabhängigkeiten. Schließlich nimmt er architekturrelevante Fehlermeldungen entgegen. Er ist auch zentraler Ansprechpartner für Organisationsrollen, wie z. B. Betrieb, Sicherheitsexperte und ähnliche.