Kapitel 5. Identifizierung architektonischer Merkmale

Um eine Architektur zu erstellen oder die Gültigkeit einer bestehenden Architektur zu bestimmen, müssen Architekten zwei Dinge analysieren: die Architektureigenschaften und die Domäne. Wie du in Kapitel 4 gelernt hast, ist es für die Ermittlung der richtigen architektonischen Merkmale ("Fähigkeiten") für ein bestimmtes Problem oder eine bestimmte Anwendung nicht nur notwendig, das Problem in der Domäne zu verstehen, sondern auch mit den Stakeholdern zusammenzuarbeiten, um festzustellen, was aus der Perspektive der Domäne wirklich wichtig ist.

Es gibt mindestens drei Ansatzpunkte, um herauszufinden, welche architektonischen Merkmale ein Projekt braucht: die Belange der Domäne, die Projektanforderungen und dein implizites Domänenwissen. Zusätzlich zu den allgemeinen impliziten Architektureigenschaften, wie Sicherheit und Modularität, haben wir festgestellt, dass es in einigen Bereichen auch implizite Architektureigenschaften gibt. Ein Architekt, der an medizinischer Software arbeitet, die von Diagnosegeräten ausgelesen wird, sollte zum Beispiel bereits wissen, wie wichtig die Datenintegrität ist und welche Folgen der Verlust von Nachrichten

haben kann. Architekten, die in diesem Bereich arbeiten, verinnerlichen die Datenintegrität, so dass sie in jeder Lösung, die sie entwerfen, implizit enthalten ist.

Extrahieren von architektonischen Merkmalenaus Domain Concerns

Die meisten architektonischen Merkmale entstehen dadurch, dass man den wichtigsten Interessengruppen zuhört und mit ihnen zusammenarbeitet, um herauszufinden, was aus geschäftlicher Sicht wichtig ist. Das mag einfach klingen, aber das Problem ist, dass Architekten und Interessengruppen unterschiedliche Sprachen sprechen. Architekten sprechen über Skalierbarkeit, Interoperabilität, Fehlertoleranz, Lernfähigkeit und Verfügbarkeit, während die Interessenvertreter der Fachbereiche über Fusionen und Übernahmen, Benutzerzufriedenheit, Markteinführungszeiten und Wettbewerbsvorteile sprechen. Das Problem ist, dass der Architekt und der Stakeholder sich nicht verstehen, weil sie sich in der Übersetzung verirrt haben. Architekten haben keine Ahnung, wie sie eine Architektur erstellen sollen, die die Nutzerzufriedenheit unterstützt, und die Stakeholder des Fachbereichs verstehen nicht, warum sich Architekten so sehr auf die Verfügbarkeit, Interoperabilität, Lernfähigkeit und Fehlertoleranz der Anwendung konzentrieren.

Glücklicherweise ist es möglich, Domain Concerns in die Sprache der architektonischen Merkmale zu "übersetzen". Tabelle 5-1 zeigt einige der

häufigsten Domänenanliegen und die entsprechenden "-ilities", die sie unterstützen. Wenn ein Architekt die wichtigsten Ziele der Domäne versteht, kann er diese Anliegen in "-ilities" übersetzen, die dann die Grundlage für korrekte und vertretbare Architekturentscheidungen bilden. Ist zum Beispiel die Markteinführungszeit wichtiger als die Skalierbarkeit, oder sollte der Architekt sich auf Fehlertoleranz, Sicherheit oder Leistung konzentrieren? Vielleicht benötigt das System all diese Eigenschaften in Kombination.

Tabelle 5-1. Übersetzen von Domänenbelangen in Architekturmerkmale

Bereich Sorge	Architektonische Merkmale
Fusionen und Übernahmen	Interoperabilität, Skalierbarkeit, Anpassbarkeit, Erweiterbarkeit
Zeit bis zur Markteinführung	Agilität, Testbarkeit, Einsatzfähigkeit
Zufriedenheit der Nutzer	Leistung, Verfügbarkeit, Fehlertoleranz, Testbarkeit, Einsatzfähigkeit, Agilität, Sicherheit
Wettbewerbsvorteil	Agilität, Testbarkeit, Einsatzfähigkeit, Skalierbarkeit, Verfügbarkeit, Fehlertoleranz
Zeit und Budget	Einfachheit, Machbarkeit

Architektonische Merkmale von Verbundwerkstoffen

Eine häufige Falle bei der Übersetzung von fachlichen Belangen in architektonische Merkmale sind falsche Gleichsetzungen, wie z. B. die

Gleichsetzung von *Agilität* mit der *Markteinführungszeit*. Agilität ist ein Beispiel für ein *zusammengesetztes* architektonisches Merkmal: ein Merkmal, für das es keine objektive Definition gibt, sondern das aus anderen messbaren Faktoren besteht. Es gibt kein Maß für Agilität, also müssen sich Architekten fragen: Woraus setzt sich Agilität zusammen? Sie umfasst Dinge wie *Einsatzfähigkeit*, *Modularität* und *Testbarkeit*, die alle messbar sind.

Ein häufiges Verhaltensmuster entsteht, wenn sich Architekten aus Bequemlichkeit nur auf einen Teil eines Bauwerks konzentrieren. Das ist so, als ob man vergisst, das Mehl in den Kuchenteig zu geben. Ein Stakeholder aus der Domäne könnte zum Beispiel sagen: "Aufgrund gesetzlicher Vorschriften ist es absolut notwendig, dass wir die Preise für die Fonds am Ende des Tages pünktlich festlegen.

Ein ineffektiver Architekt könnte darauf reagieren, indem er sich ausschließlich auf die Leistung konzentriert, da dies der Hauptfokus des Bereichs zu sein scheint. Dieser Ansatz wird jedoch aus vielen Gründen fehlschlagen:

- Es spielt keine Rolle, wie schnell das System ist, wenn es nicht verfügbar ist, wenn es gebraucht wird.
- Wenn der Bereich wächst und mehr Fonds angelegt werden, muss das System skalierbar sein, um die Tagesendverarbeitung rechtzeitig abzuschließen.
- Das System muss nicht nur verfügbar, sondern auch zuverlässig sein, damit es nicht abstürzt, wenn die Fondspreise am Tagesende berechnet werden.

- Was passiert, wenn das System abstürzt, während die Preisermittlung am Tagesende zu 85% abgeschlossen ist? Es muss in der Lage sein, sich wieder zu erholen und die Preisbildung dort fortzusetzen, wo es aufgehört hat.
- Schließlich mag das System schnell sein, aber werden die Fondspreise auch richtig berechnet?

Neben der Leistung muss der Architekt also auch auf Verfügbarkeit, Skalierbarkeit, Zuverlässigkeit, Wiederherstellbarkeit und Überprüfbarkeit achten.

Viele Geschäftsziele beginnen als zusammengesetzte Architekturmerkmale. Sie zu zerlegen und den daraus resultierenden Merkmalen objektive Definitionen zu geben, ist Teil der Aufgabe der Architekten. (Wie wichtig das ist, werden wir in Kapitel 6 sehen.)

Architektonische Merkmale extrahieren

Die meisten architektonischen Merkmale stammen aus expliziten Aussagen in einer Art Anforderungsdokument. Zum Beispiel enthalten Listen von Domänenbelangen in der Regel explizite Zahlen über die erwarteten Nutzer und den Umfang. Andere Merkmale ergeben sich aus dem inhärenten Fachwissen der Architekten (einer der vielen Gründe, warum jeder Architekt sein Fachgebiet kennen sollte).

Angenommen, du bist Architekt und entwirfst eine Anwendung, mit der du dich für einen Kurs an einer Universität anmelden kannst. Um die Rechnung zu vereinfachen, nimmst du an, dass die Schule 1.000 Schüler/innen hat und 10 Stunden für die Anmeldung benötigt. Solltest du davon ausgehen, dass sich die Schüler/innen gleichmäßig auf diese 10 Stunden verteilen, und das System so gestalten, dass es eine gleichmäßige Verteilung ermöglicht? Oder solltest du auf der Grundlage deiner Kenntnisse über die Gewohnheiten und Neigungen der Schüler/innen ein System entwickeln, das alle 1.000 Schüler/innen, die sich in den letzten 10 Minuten anmelden wollen, bewältigen kann?

Jeder, der weiß, wie sehr Studierende stereotypisch prokrastinieren, kennt die Antwort auf diese Frage! Solche Details tauchen in Anforderungsdokumenten nur selten auf - und doch fließen sie in die Entwurfsentscheidungen ein.

DER URSPRUNG DER ARCHITEKTUR KATAS

Vor mehr als einem Jahrzehnt entwickelte Ted Neward, ein bekannter Architekt, *Architektur-Katas*, eine clevere Methode, mit der angehende Architekten üben können, architektonische Merkmale aus zielgerichteten Beschreibungen abzuleiten. Das Wort *Kata* stammt aus dem Japanischen und bezeichnet eine einzelne Übung aus dem Kampfsport, bei der es auf die richtige Form und Technik ankommt.

> *Wie kommen wir zu guten Designern? Gute Designer entwerfen natürlich.*
>
> —Fred Brooks

Große Architekturprojekte brauchen Zeit, und es ist üblich, dass Architekten in ihrer Karriere vielleicht ein halbes Dutzend Systeme entwerfen. Wie sollen wir also gute Architekten bekommen? Der Schlüssel ist, angehenden Architekten die Möglichkeit zu geben, ihr Handwerk zu üben. Um einen Lehrplan zu erstellen, hat Ted die erste Website für Architektur-Katas erstellt, die eure Autoren Neal und Mark auf der begleitenden Website zu diesem Buch unter *fundamentalsofsoftwarearchitecture.com* angepasst und aktualisiert haben. Die Architektur-Katas bleiben ihrem ursprünglichen Zweck treu und sind ein nützliches Labor für angehende Architekten.

Arbeiten mit Katas

Die Grundvoraussetzung ist, dass jede Kata-Übung ein Problem in Form von Fachbegriffen, eine Reihe von Anforderungen und einen zusätzlichen Kontext (Dinge, die nicht in den Anforderungen vorkommen, aber den Entwurf beeinflussen könnten) liefert. Kleine Teams arbeiten für eine bestimmte Zeit an einem Entwurf (bestehend aus einer Analyse der architektonischen Merkmale und Diagrammen). Dann tauschen die Gruppen ihre Ergebnisse aus und stimmen darüber ab, wer die beste Architektur entwickelt hat.

Jede Kata hat vordefinierte Abschnitte:

Beschreibung
Das allgemeine Problem, das das System lösen soll.

Benutzer
Die erwartete Anzahl und/oder Art der Nutzer des Systems.

Anforderungen
Anforderungen des Fachgebiets (wie sie von Nutzern und Experten des Fachgebiets erwartet werden).

Zusätzlicher Kontext
Die Autoren haben das Kata-Format auf der zuvor erwähnten Website mit einem *zusätzlichen Kontextabschnitt* mit wichtigen Überlegungen aktualisiert, wodurch die Übungen realistischer werden.

Wir ermutigen die Leser/innen, die Website zu nutzen, um ihre eigene Kata-Übung durchzuführen. Jeder kann ein Brown-Bag-Lunch

veranstalten, bei dem ein Team von angehenden Architekten ein Problem lösen kann. Ein erfahrener Architekt kann den Entwurf und die Analyse der Kompromisse bewerten und entweder vor Ort oder im Nachhinein über verpasste Kompromisse und alternative Entwürfe diskutieren. Die Entwürfe werden nicht sehr aufwendig sein, weil die Übung zeitlich begrenzt ist.

Als Nächstes werden wir eine Architektur-Kata verwenden, um zu zeigen, wie Architekten architektonische Merkmale aus den Anforderungen ableiten. Willkommen bei der *Silicon Sandwiches* Kata.

Kata: Silizium-Sandwiches

Beschreibung
Ein nationaler Sandwich-Shop will zusätzlich zu seinem derzeitigen Call-in-Service auch Online-Bestellungen ermöglichen.

Benutzer
Tausende, vielleicht eines Tages auch Millionen.

Anforderungen

- Erlaube den Nutzern, eine Bestellung aufzugeben und, wenn der Laden einen Lieferservice anbietet, die Abholung oder Lieferung auszuwählen.
- Gib den Abholern eine Uhrzeit für die Abholung ihrer Bestellung und eine Wegbeschreibung zum Geschäft (die mit verschiedenen

externen Kartendiensten mit Verkehrsinformationen integriert werden muss).

- Beim Lieferservice schickst du den Fahrer mit der Bestellung zum Nutzer.
- Biete Zugriff auf mobile Geräte.
- Biete täglich nationale Werbeaktionen und Sonderangebote an.
- Biete täglich lokale Sonderaktionen und Specials an.
- Akzeptiere die Zahlung online, im Laden oder bei der Lieferung.

Zusätzlicher Kontext

- Sandwich-Läden sind Franchise-Läden, die jeweils einen anderen Besitzer haben.
- Die Muttergesellschaft hat Pläne, in naher Zukunft nach Übersee zu expandieren.
- Das Ziel des Unternehmens ist es, billige Arbeitskräfte einzustellen, um den Gewinn zu maximieren.

Wie würdest du in diesem Szenario die Architekturmerkmale ableiten? Jeder Teil der Anforderung kann zu einem oder mehreren Aspekten der Architektur beitragen (und viele werden es nicht). Der Architekt entwirft hier nicht das gesamte System - ein beträchtlicher Teil des Aufwands muss immer noch in die Erstellung von Code fließen, um das Domänen-

Design zu lösen (siehe Kapitel 8). Stattdessen solltest du nach Dingen suchen, die das Design beeinflussen, vor allem struktureller Art.

Trenne zunächst die in Frage kommenden Architekturmerkmale in explizite und implizite Merkmale.

Explizite Merkmale

Explizite architektonische Merkmale erscheinen in einer Anforderungsspezifikation als Teil des notwendigen Designs. Eine Shopping-Website kann zum Beispiel eine bestimmte Anzahl von gleichzeitigen Nutzern unterstützen, die die Fachanalytiker in den Anforderungen angeben. Überprüfe jeden Teil der Anforderungen, um zu sehen, ob er zu einem Architekturmerkmal beiträgt. Aber zuerst solltest du die Vorhersagen auf der Domänenebene über die erwarteten Kennzahlen betrachten, wie sie im Abschnitt Benutzer der Kata dargestellt sind.

Eines der ersten Details, das dir ins Auge fallen sollte, ist die Anzahl der Nutzer/innen: derzeit Tausende, vielleicht eines Tages Millionen (dies ist ein sehr ehrgeiziger Sandwich-Shop!). Daher ist die *Skalierbarkeit* - *die*Fähigkeit, eine große Anzahl gleichzeitiger Nutzer ohne ernsthafte Leistungseinbußen zu bewältigen - eines der wichtigsten Merkmale der Architektur. Beachte, dass in der Aufgabenstellung nicht explizit nach der Skalierbarkeit gefragt wurde, sondern diese Anforderung als erwartete Anzahl von Nutzern ausgedrückt wurde. Architekten müssen die Sprache des Fachgebiets oft in technische Entsprechungen umwandeln.

Wahrscheinlich brauchst du auch *Elastizität* - *die*Fähigkeit, eine Vielzahl von Anfragen zu bearbeiten. Diese beiden Eigenschaften werden oft in einen Topf geworfen, aber sie haben unterschiedliche Voraussetzungen. Die Skalierbarkeit sieht wie das Diagramm in Abbildung 5-1 aus.

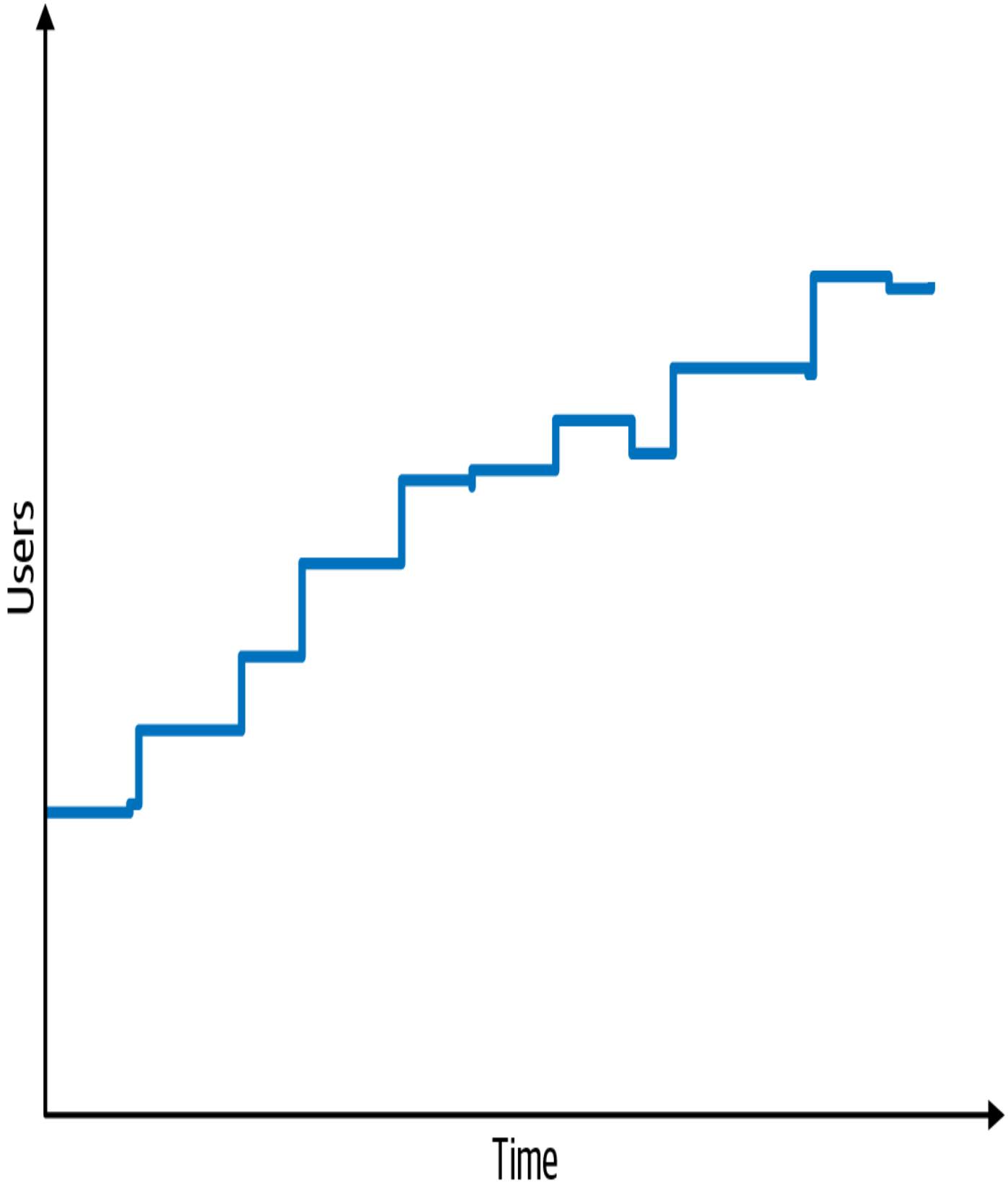

Abbildung 5-1. Skalierbarkeit ist ein Maß für den Anstieg der Nutzerzahlen im Laufe der Zeit

Die Elastizität hingegen misst Verkehrsbursts, wie in Abbildung 5-2 dargestellt.

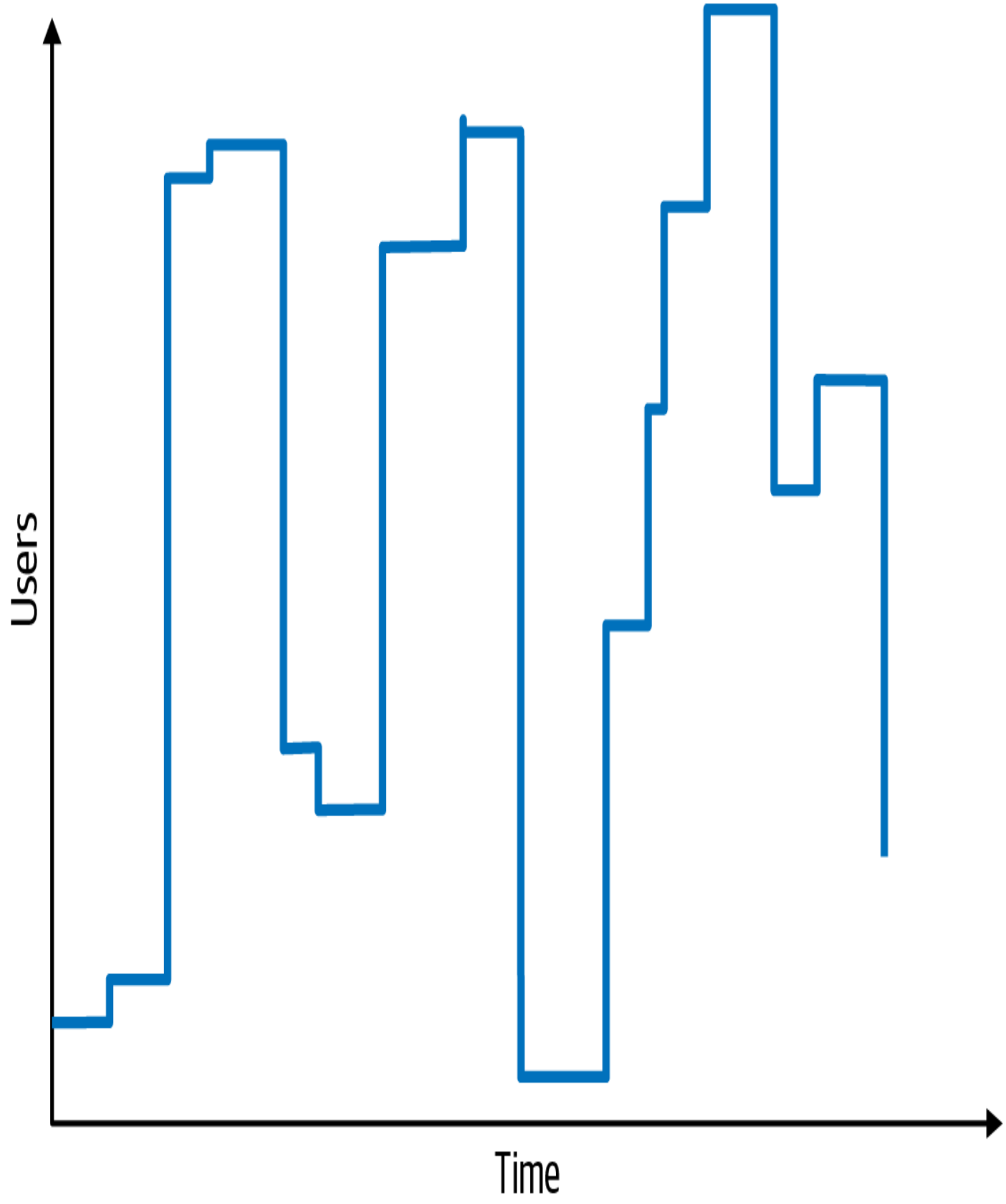

Abbildung 5-2. Elastische Systeme müssen einem Ansturm von Nutzern standhalten

Manche Systeme sind skalierbar, aber nicht elastisch. Zum Beispiel ist die Anzahl der Nutzer/innen eines Hotelreservierungssystems ohne Sonderverkäufe oder Veranstaltungen wahrscheinlich nur saisonal begrenzt. Betrachte im Gegensatz dazu ein Konzertkarten-Buchungssystem. Wenn neue Tickets in den Verkauf gehen, wird die Website von begeisterten Fans überflutet, was ein hohes Maß an Elastizität erfordert. Elastische Systeme brauchen oft auch *Skalierbarkeit*: die Fähigkeit, Stoßzeiten und eine hohe Anzahl gleichzeitiger Nutzer zu bewältigen.

Elastizität kommt in den Anforderungen von Silicon Sandwiches nicht vor, aber du solltest sie trotzdem als wichtigen Aspekt identifizieren. In den Anforderungen werden manchmal direkt architektonische Merkmale genannt, aber andere sind in der Problemdomäne versteckt. Ist der Verkehr eines Sandwich-Ladens den ganzen Tag über gleichmäßig, oder gibt es zu den Essenszeiten besonders viele Besucher? Mit ziemlicher Sicherheit ist Letzteres der Fall, also ermittle dieses potenzielle Architekturmerkmal.

Als Nächstes betrachtest du jede dieser Geschäftsanforderungen der Reihe nach, um zu sehen, ob sie architektonischeMerkmale erfordert:

> *Du gibst deine Bestellung auf und wählst, wenn der Shop einen Lieferservice anbietet, die Abholung oder Lieferung aus.*
> Es scheinen keine besonderen architektonischen Merkmale notwendig zu sein, um diese Anforderung zu erfüllen.

Kunden, die ihr Sandwich abholen, erhalten eine Uhrzeit und eine Wegbeschreibung zum Laden (die die Möglichkeit bieten muss, externe Kartendienste mit Verkehrsinformationen zu integrieren).
Externe Mapping-Dienste implizieren Integrationspunkte, die sich auf Aspekte wie die Zuverlässigkeit auswirken können. Ein Beispiel: Ein Entwickler baut ein System, das auf ein System eines Drittanbieters angewiesen ist. Wenn diese Aufrufe fehlschlagen, wirkt sich das auf die Zuverlässigkeit des aufrufenden Systems aus. Umgekehrt solltest du dich davor hüten, die architektonischen Merkmale zu sehr zu spezifizieren. Was ist, wenn der externe Verkehrsdienst ausfällt? Soll die Silicon Sandwiches Seite fehlschlagen oder soll sie ohne Verkehrsinformationen nur etwas weniger effizient sein? Architekten sollten sich immer davor hüten, unnötige Brüchigkeit oder Anfälligkeit in ihre Entwürfe einzubauen.

Beim Lieferservice schickst du den Fahrer mit der Bestellung zum Nutzer.
Es scheinen keine besonderen architektonischen Merkmale notwendig zu sein, um dieseAnforderung zu erfüllen.

Ermögliche den Zugriff auf mobile Geräte.
Diese Anforderung wirkt sich in erster Linie auf die Benutzerfreundlichkeit der Anwendung aus und deutet darauf hin, dass entweder eine mobile Webanwendung oder mehrere native Webanwendungen entwickelt werden müssen. Angesichts des begrenzten Budgets und der Einfachheit der Anwendung wäre es wahrscheinlich ein Overkill, mehrere Anwendungen zu erstellen,

daher deutet das Design auf eine mobil-optimierte Webanwendung hin. Daher solltest du bestimmte leistungsbezogene Architekturmerkmale festlegen, um die Ladezeit der Seiten und andere mobilitätsrelevante Merkmale zu optimieren.

In solchen Situationen solltest du nicht allein handeln. Arbeite mit UX-Designern, Stakeholdern und anderen interessierten Parteien zusammen, um solche Entscheidungen zu überprüfen. Es kann sein, dass das Unternehmen ein bestimmtes Verhalten verlangt, das nur möglich ist, wenn du native Anwendungen entwickelst.

Biete täglich nationale Sonderaktionen und Specials an; biete täglich lokale Sonderaktionen und Specials an.
Diese beiden Anforderungen erfordern eine individuelle Anpassung von Aktionen und Angeboten. Anforderung 1 beinhaltet auch individuelle Verkehrsinformationen auf der Grundlage der Adresse des Nutzers. Ausgehend von diesen beiden Anforderungen könntest du die Anpassungsfähigkeit als architektonisches Merkmal in Betracht ziehen. Der Mikrokernel-Architekturstil (siehe Kapitel 13) unterstützt zum Beispiel ein individuelles Verhalten sehr gut, indem er eine Plug-in-Architektur definiert. In diesem Fall ist das Standardverhalten im Kern enthalten, und die Entwickler schreiben die optionalen, ortsabhängigen benutzerdefinierten Teile als Plug-ins. Aber auch ein traditionelles Design kann diese Anforderung mit Hilfe von Entwurfsmustern (wie der Template-Methode) erfüllen. Dieses Problem ist in der Architektur häufig anzutreffen und erfordert von den Architekten ein ständiges Abwägen von Kompromissen zwischen

konkurrierenden Optionen. In "Design versus Architektur und Kompromisse" gehen wir näher auf bestimmte Kompromisse ein .

Akzeptiere die Zahlung online, im Laden oder bei der Lieferung.
Online-Zahlungen erfordern ein gewisses Maß an Sicherheit, aber nichts in dieser Anforderung deutet auf ein besonders hohes Sicherheitsniveau hin, das über das hinausgeht, was implizit ist. Architekten können die Sicherheit entweder mit dem Design oder mit der Architektur behandeln, so dass sie bei dieser Anwendung nur ein minimales architektonisches Merkmal darstellt.

Sandwich-Läden sind Franchise-Läden, die jeweils einen anderen Besitzer haben.
Diese Anforderung kann zu Kostenbeschränkungen für die Architektur führen. Überprüfe die Machbarkeit anhand von Einschränkungen wie Kosten, Zeit und Personalqualifikationen, um zu sehen, ob eine einfache oder opferbereite Architektur gerechtfertigt ist.

Die Muttergesellschaft hat Pläne, in naher Zukunft nach Übersee zu expandieren.
Diese Anforderung impliziert *Internationalisierung* (oft als "i18n" abgekürzt). Es gibt viele Designtechniken, um diese Anforderung zu erfüllen, die keine besondere Struktur erfordern sollten. Allerdings wird dies sicherlich die UX-Entscheidungen beeinflussen.

Das Ziel des Unternehmens ist es, billige Arbeitskräfte einzustellen, um den Gewinn zu maximieren.

> Diese Anforderung legt nahe, dass die Benutzerfreundlichkeit wichtig ist, aber auch hier geht es mehr um das Design als um architektonische Merkmale.

Das dritte architektonische Merkmal, das wir aus den vorangegangenen Anforderungen ableiten können, ist die *Leistung*: Niemand möchte bei einem Sandwich-Shop kaufen, der eine schlechte Leistung hat, vor allem nicht zu Spitzenzeiten. *Leistung* ist jedoch ein differenziertes Konzept. Welche *Art von* Leistung solltest du anstreben? (Wir behandeln die verschiedenen Nuancen der Leistung in Kapitel 6).

Es ist auch wichtig, Leistungszahlen in Verbindung mit Skalierbarkeitszahlen zu definieren. Mit anderen Worten: Lege einen Grundwert für die Leistung ohne eine bestimmte Skalierung fest und bestimme ein akzeptables Leistungsniveau für eine bestimmte Anzahl von Nutzern. Oft stehen architektonische Merkmale in Wechselwirkung zueinander, so dass die Architekten gezwungen sind, sie in Relation zueinander zu definieren.

Implizite Merkmale

Viele architektonische Merkmale werden in den Anforderungsdokumenten nicht genannt, obwohl sie wichtige Aspekte des Designs sind. Ein implizites architektonisches Merkmal, das das System unterstützen sollte, ist die *Verfügbarkeit*: Es muss sichergestellt werden, dass die Nutzer/innen auf die Website zugreifen können. Eng mit der Verfügbarkeit verbunden ist die *Stabilität*: Es muss sichergestellt

werden, dass die Website während der Interaktionen verfügbar bleibt. Niemand möchte auf einer Website einkaufen, die die Verbindung unterbricht und ihn zwingt, sich erneut einzuloggen.

Sicherheit erscheint als implizites Merkmal in jedem System: Niemand will unsichere Software erstellen. Je nachdem, wie kritisch sie ist, kann man ihr jedoch unterschiedliche Priorität einräumen, was die Verzahnung unserer Definition verdeutlicht. Du kannst Sicherheit als architektonisches Merkmal betrachten, wenn sie einen strukturellen Aspekt des Entwurfs beeinflusst und für die Anwendung kritisch oder wichtig ist.

Bei Silicon Sandwiches kannst du davon ausgehen, dass die Zahlungen von einer dritten Partei abgewickelt werden sollten. Solange die Entwickler die allgemeine Sicherheitshygiene beachten (Kreditkartennummern nicht im Klartext weitergeben, nicht zu viele Informationen speichern usw.), reicht ein gutes Design der Anwendung aus; du brauchst keine besondere Struktur, um die Sicherheit zu gewährleisten. Erinnere dich daran, dass architektonische Merkmale *synergetisch*sind - *jedes*architektonische Merkmal steht in Wechselwirkung mit den anderen. Deshalb ist die Überspezifikation von Architekturmerkmalen eine so häufige Falle. Eine Überspezifizierung ist genauso schädlich wie eine Unterspezifizierung, weil sie den Systementwurf zu sehr verkompliziert.

Die letzte wichtige architektonische Eigenschaft, die Silicon Sandwiches unterstützen muss, die *Anpassbarkeit*, setzt sich aus mehreren Details der Anforderungen zusammen. Mehrere Teile der Problemdomäne

bieten benutzerdefiniertes Verhalten: Rezepte, lokale Verkäufe und Richtungen, die lokal außer Kraft gesetzt werden können. Daher sollte die Architektur benutzerdefiniertes Verhalten unterstützen. Normalerweise würde dies unter Anwendungsdesign fallen. Wenn jedoch ein Teil der Problemdomäne auf eine benutzerdefinierte Struktur angewiesen ist, um sie zu unterstützen, fällt dies laut unserer Definition in den Bereich der architektonischen Merkmale. Dieses Designelement ist jedoch nicht entscheidend für den Erfolg der Anwendung. Erinnere dich daran, dass es bei der Auswahl von Architekturmerkmalen keine richtigen Antworten gibt, sondern nur falsche - oder:

> *In der Architektur gibt es keine falschen Antworten, nur teure.*
>
> —Eines der berühmten Zitate von Markus

DESIGN VS. ARCHITEKTUR UND KOMPROMISSE

In der Silicon Sandwiches Kata würdest du wahrscheinlich die Anpassungsfähigkeit als Teil des Systems identifizieren, aber dann stellt sich die Frage: Architektur oder Design? Die Architektur impliziert eine strukturelle Komponente, während das Design innerhalb der Architektur angesiedelt ist. Für Silicon Sandwiches könntest du einen Architekturstil wie den Mikrokernel wählen und die strukturelle Unterstützung für die Anpassungsfähigkeit aufbauen. Wenn du dich jedoch aufgrund konkurrierender Interessen für einen anderen Stil entscheidest, können die Entwickler die Anpassung mit dem Entwurfsmuster Template Method umsetzen, das es übergeordneten Klassen ermöglicht, Workflows zu definieren, die in untergeordneten Klassen überschrieben werden können. Welche Option ist besser?

Wie alles in der Architektur hängt es davon ab. Erstens: Gibt es gute Gründe, *keine* Mikrokernel-Architektur zu implementieren (z. B. Leistung und Kopplung)? Zweitens: Sind andere wünschenswerte architektonische Merkmale in dem einen Entwurf schwieriger als in dem anderen? Drittens: Wie viel würde es kosten, alle architektonischen Merkmale in der Architektur und im Design zu unterstützen? Diese Art der Analyse von Kompromissen ist ein wichtiger Teil der Aufgabe eines Architekten.

Vor allem ist es wichtig, mit den Entwicklern, dem Projektmanager, dem Betriebsteam und anderen Mitgestaltern des Softwaresystems zusammenzuarbeiten. Architekturentscheidungen sollten nicht isoliert vom Implementierungsteam getroffen werden (was zu dem

gefürchteten Architekturmuster des Elfenbeinturms führt). Im Fall von Silicon Sandwiches sollten der Architekt, der technische Leiter, die Entwickler und die Domänenanalytiker zusammenarbeiten, um zu entscheiden, wie die Anpassungsfähigkeit am besten umgesetzt werden kann.

Du solltest dir nicht zu viele Gedanken darüber machen, wie du die richtige Architektur findest. Entwickler können Funktionen auf verschiedene Arten implementieren. Die richtige Identifizierung wichtiger Strukturelemente kann jedoch ein einfacheres oder eleganteres Design ermöglichen. Du könntest eine Architektur entwerfen, die die Anpassungsfähigkeit strukturell nicht berücksichtigt und stattdessen verlangt, dass das Design der Anwendung selbst dieses Verhalten unterstützt (siehe "Design vs. Architektur und Kompromisse"). Erinnere dich: In der Architektur gibt es kein bestes Design, sondern nur eine Sammlung von Kompromissen, die am wenigsten schlecht sind.

Priorisiere diese architektonischen Merkmale, wenn du versuchst, die einfachsten erforderlichen Sets zu finden. Wenn das Team einen ersten Versuch unternommen hat, die architektonischen Merkmale zu bestimmen, ist es sinnvoll, das am wenigsten wichtige Merkmal zu ermitteln. Wenn du eines ausschließen müsstest, welches wäre das? Im Allgemeinen streichen Architekten eher die expliziten architektonischen Merkmale, da viele der impliziten Merkmale den allgemeinen Erfolg unterstützen. Die Art und Weise, wie wir definieren, was kritisch oder wichtig für den Erfolg ist, hilft dabei zu bestimmen, ob die Anwendung

wirklich jedes architektonische Merkmal *benötigt*. Der Versuch, das am wenigsten zutreffende Merkmal zu bestimmen, kann dir helfen, die kritische Notwendigkeit zu ermitteln.

Welches der architektonischen Merkmale, die wir identifiziert haben, ist im Fall von Silicon Sandwiches am wenigsten wichtig? Auch hier gibt es keine absolut richtige Antwort. In diesem Fall könnte die Lösung jedoch entweder an Anpassungsfähigkeit oder an Leistung verlieren. Wir könnten die Anpassungsfähigkeit als architektonisches Merkmal streichen und planen, dieses Verhalten als Teil des Anwendungsdesigns zu implementieren. Von den operativen Architektureigenschaften ist die Leistung wahrscheinlich am wenigsten entscheidend für den Erfolg. Das bedeutet natürlich nicht, dass die Entwickler eine Anwendung mit schlechter Leistung entwickeln wollen; es bedeutet nur, dass die Leistung bei diesem Entwurf nicht wichtiger ist als andere Merkmale wie Skalierbarkeit oder Verfügbarkeit.

Einschränkung und Priorisierung von architektonischen Merkmalen

Ein Tipp für die Zusammenarbeit mit den Stakeholdern der Domäne bei der Definition der wichtigsten architektonischen Merkmale: Bemühe dich, die endgültige Liste so kurz wie möglich zu halten. Ein häufig anzutreffendes Verhaltensmuster ist der Versuch, eine *generische Architektur* zu entwerfen: eine Architektur, die *alle* Architekturmerkmale unterstützt. Jedes Merkmal, das die Architektur

unterstützt, verkompliziert das gesamte Systemdesign, so dass die Unterstützung zu vieler Architekturmerkmale zu immer größerer Komplexität führt. Und das, bevor der Architekt und die Entwickler überhaupt angefangen haben, sich mit der Problemdomäne zu befassen - der ursprünglichen Motivation für das Schreiben der Software. Kümmere dich nicht um die Anzahl der Merkmale, sondern erinnere dich an die Motivation: den Entwurf einfach zu halten.

FALLSTUDIE: DIE *VASA*

Das Paradebeispiel für eine Überspezifizierung architektonischer Merkmale, die ein Projekt letztendlich zum Scheitern bringt, ist die *Vasa*. Sie war ein schwedisches Kriegsschiff, das zwischen 1626 und 1628 von einem König gebaut wurde, der das prächtigste Schiff aller Zeiten haben wollte. Bis zu diesem Zeitpunkt waren Schiffe entweder Truppentransporter oder Kanonenboote - die *Vasa* war beides. Die meisten Schiffe hatten ein Deck - die *Vasa* hatte zwei! Alle Kanonen waren doppelt so groß wie die auf ähnlichen Schiffen. Trotz einiger Bedenken konnten die Schiffbauexperten König Adolphus nicht widersprechen.

Um den Abschluss des Baus zu feiern, fuhr die *Vasa* in den Stockholmer Hafen und schoss eine Kanonensalve von einer Seite ab. Da das Schiff kopflastig war, kenterte es leider und sank auf den Grund. Im Jahr 1961 wurde das Schiff geborgen und befindet sich heute in einem Museum in Stockholm.

Die Zusammenarbeit zwischen dem Architekten und den Stakeholdern ist bei der Analyse der Architektureigenschaften entscheidend. Wenn der Architekt jedoch eine umfangreiche Liste möglicher Architektureigenschaften vorlegt und die Stakeholder fragt, welche Eigenschaften die Architektur unterstützen soll, was wird die Antwort jedes Mal sein? *"Alle!"*

Daher brauchen Architekten eine Methode, um zu bestimmen, welche architektonischen Merkmale die strukturellen Entscheidungen beeinflussen und für den Erfolg entscheidend sind. Die Autoren haben eine Reihe von Techniken entwickelt, um diese Arbeit zu erleichtern, darunter ein "Arbeitsblatt" für architektonische Merkmale, das in Abbildung 5-3 dargestellt ist (und unter *https://developertoarchitect.com/resources.html* heruntergeladen werden kann *).*

Das Arbeitsblatt ist für eine interaktive Sitzung gedacht, die von einem Architekten geleitet wird, der die Meinungen der Beteiligten zur Anzahl und den Details der gewünschten architektonischen Merkmale einholt. Auf der linken Seite siehst du die sieben Felder, in denen du die gewünschten architektonischen Merkmale auflisten kannst.

Warum sieben? Warum nicht? Im Ernst, ein Architekt muss die Liste auf eine vernünftige Zahl beschränken - sechs oder acht würden auch funktionieren (obwohl es eine interessante Psychologie hinter der Zahl sieben gibt). Die zweite Spalte enthält einige implizite architektonische Merkmale. Diese kommen in den meisten Systemen vor, aber manchmal werden sie von den Architekten als Hauptanliegen der Anwendung

eingestuft, die eine besondere Gestaltung und Berücksichtigung erfordern. In diesen Fällen "zieht" der Architekt die impliziten Merkmale in die erste Spalte. Wenn sich die erste Spalte füllt und ein besseres Merkmal ein bestehendes verdrängt, verschiebt der Architekt es in die Kategorie "Andere in Betracht gezogene Merkmale".

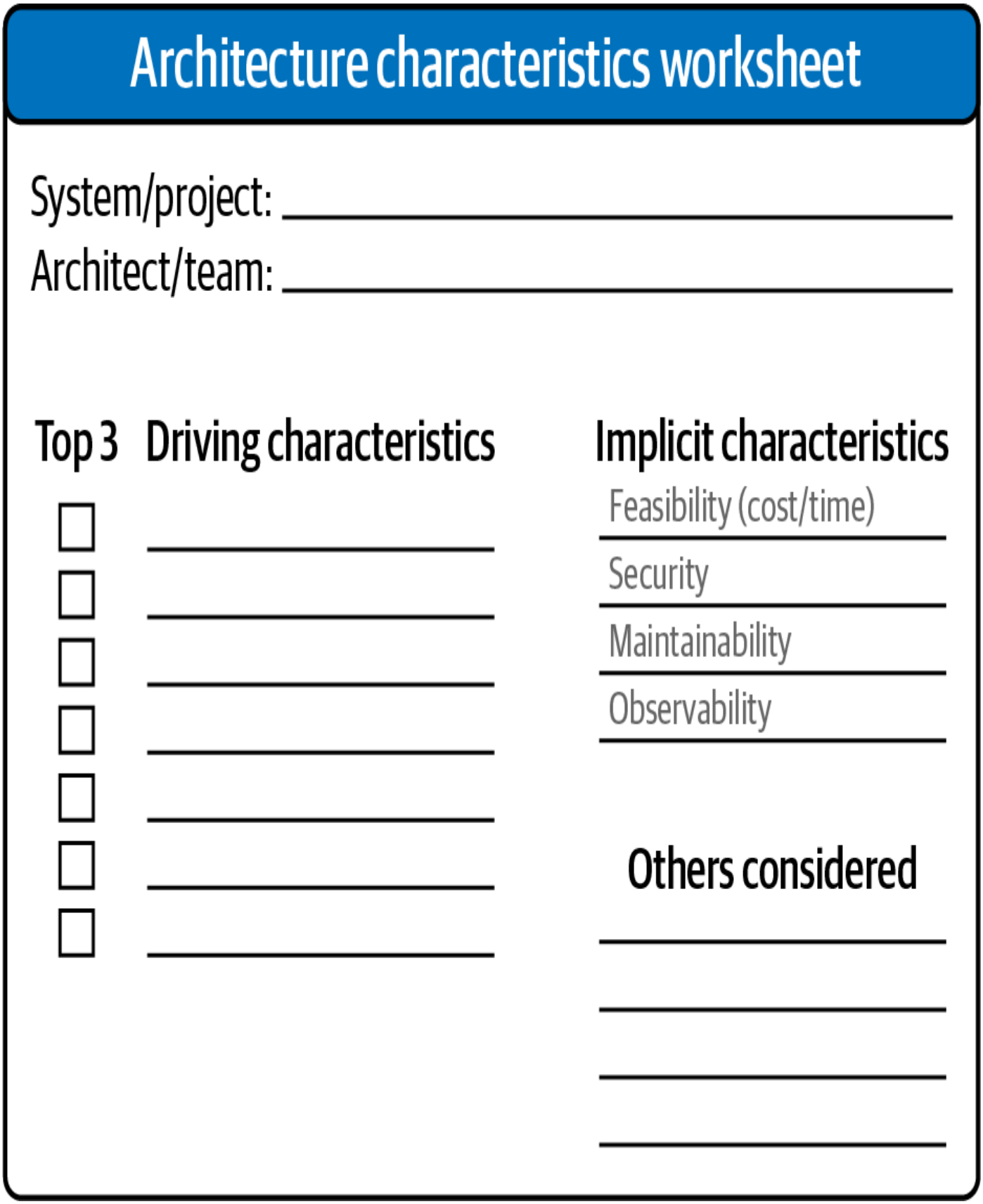

Abbildung 5-3. Arbeitsblatt "Architektonische Merkmale

Der letzte Schritt besteht darin, gemeinsam die drei wichtigsten architektonischen Merkmale in beliebiger Reihenfolge auszuwählen (kreuze das Kästchen neben jedem Merkmal an). Auf diese Weise können die Architekten eine kurze, nach Prioritäten geordnete Liste der wichtigsten Faktoren erstellen, die sie für ihre Entwurfsentscheidungen und die Analyse von Kompromissen nutzen können.

Viele Architekten und Stakeholder wollen eine endgültige Liste von Architekturmerkmalen erstellen, die die Anwendung oder das System unterstützen muss. Das ist zwar ein wünschenswertes Ergebnis, aber in den meisten Fällen ist es ein Irrweg, der nicht nur Zeit kostet, sondern auch zu unnötiger Frustration und Meinungsverschiedenheiten mit den wichtigsten Interessengruppen führt. Selten werden sich alle Beteiligten über die Priorität jedes einzelnen Merkmals einig sein. Ein besserer Ansatz ist es, die drei wichtigsten Merkmale aus der endgültigen Liste auszuwählen (in beliebiger Reihenfolge). Das macht es viel einfacher, einen Konsens zu finden und fördert die Diskussion darüber, was am wichtigsten ist. All dies hilft dem Architekten bei der Analyse von Kompromissen, wenn er wichtige architektonische Entscheidungen trifft
.