Guido Oelmann

Modularisierung mit Java 9

Grundlagen und Techniken für langlebige Softwarearchitekturen

dpunkt.verlag

Dipl.-Inform. **Guido Oelmann** arbeitet seit vielen Jahren als selbstständiger Softwarearchitekt und Berater. Zu seinen Schwerpunkten gehören neben agilen Entwicklungsmethoden und modernen Softwarearchitekturen, der Einsatz von Java-/Java-EE-Technologien in verteilten Systemen. Er unterstützt Unternehmen bei der Durchführung ihrer Projekte und verfügt über jahrelange Erfahrung beim Entwurf und der Entwicklung großer IT-Systeme in unterschiedlichen Branchen. Darüber hinaus gibt er sein Wissen regelmäßig bei Vorträgen, in Artikeln und Schulungen weiter.

Guido Oelmann

Modularisierung mit Java 9

Grundlagen und Techniken für langlebige Softwarearchitekturen

dpunkt.verlag

Guido Oelmann
Guido.Oelmann@JavaAkademie.de

Lektorat: René Schönfeldt
Projektmanagement: Miriam Metsch
Copy-Editing: Sandra Gottmann, Münster-Nienberge
Satz und Herstellung: Da-TeX, Leipzig
Umschlaggestaltung: Helmut Kraus, www.exclam.de
Druck und Bindung: Media-Print Informationstechnologie, Paderborn

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

ISBN:
Print 978-3-86490-477-6
PDF 978-3-96010-180-2
ePub 978-3-96010-181-9
mobi 978-3-96010-182-6

1. Auflage 2018

Wieblinger Weg 17
69123 Heidelberg

5 4 3 2 1 0

Vorwort

Software ist heutzutage allgegenwärtig – sichtbar und unsichtbar. Es existiert kaum ein Bereich des Lebens, der nicht auf die ein oder andere Weise von Bit und Bytes durchdrungen ist. Angefangen bei der simplen Bezahlung im Lebensmittelgeschäft, wo softwarebasierte Kassensysteme zum Einsatz kommen, über das allgegenwärtige Smartphone hin zu WebShops und verteilten Unternehmensanwendungen zur Abbildung von Geschäftsprozessen. Mit dem Internet der Dinge (*engl. Internet of Things (IoT)*) wird sich dieser Trend verstärken und den sogenannten Digitalisierungsprozess unserer Gesellschaft weiter vorantreiben. Überall fließen Daten, die verarbeitet werden müssen, und dadurch erhöhen sich die Anforderungen an die Software. Die Folge sind immer größer und mächtiger werdende Softwaresysteme mit zunehmend komplexerem Charakter. Daraus ergeben sich neue Probleme und Herausforderungen, die es zu bewältigen gilt. Monolithische Systeme stoßen hier zunehmend an ihre Grenzen.

Um diesem Wandel in der Softwareentwicklung gerecht zu werden, hat sich in den letzten Jahren viel getan. Die agile Softwareentwicklung ist mittlerweile in vielen Unternehmen etabliert und Themen wie Configuration Management (CM) und Continuous Delivery (CD) sind gelebte Disziplinen. Und auch auf organisatorischer Ebene bringt die Stellenschaffung von DevOps als Erweiterung des agilen Entwicklungsprozesses einen großen, positiven Einfluss. Darüber hinaus zeigten der Aufstieg von Microservices und Self Contained Systems als Architekturmuster und insbesondere die Container-Technologie, dass sich die Art und Weise, wie Software entwickelt und veröffentlicht wird, ändert. Dies bringt größere Flexibilität, Produktivität und eine höhere Geschwindigkeit bei der Veröffentlichung von Software mit sich, aber auch Mehraufwände im Management und neue Anforderungen im Bereich des automatisierten Testens und Monitorings. Softwareentwicklung ist aktuell und auch in Zukunft weiterhin ein hochkomplexer Prozess, der in seiner Gesamtheit über den reinen Entwurf und die Implementierung hinausgeht.

Umso wichtiger ist die Reduzierung von Komplexität innerhalb solcher Softwaresysteme, um diese auch in den Phasen des Betriebs, der Wartung und Weiterentwicklung unter Kontrolle zu halten. Die Modularisierung, mit der sich das vorliegende Buch beschäftigt, ist dabei ein wichtiger Baustein. Und Java 9 bietet nun eine direkte Unterstützung für den Modularisierungsprozess.

In diesem Sinne viel Spaß beim Lesen des Buches und der Entwicklung von Software.

Guido Oelmann
Dortmund, 12. September 2017
www.Java-Akademie.de

Überblick über das Buch

Das Buch behandelt die Modularisierung mit Java. Neben der Darstellung des Prinzips der Modularisierung und warum dieses wichtig ist, wird konkret gezeigt, wie mit Java eigene Module erstellt werden können und wie eine Applikation basierend auf Modulen gebaut wird. Das neue modularisierte JDK wird vorgestellt und dessen Wichtigkeit für die Anwendung im Internet der Dinge. Daneben wird ein Vergleich zum bisherigen quasi Modularisierungsstandard OSGi gezogen. Es werden weitergehende Themen wie die Migration von Projekten hin zu Modulen und die Nutzung von Modulen im Kontext moderner Softwareentwicklung behandelt. Letzteres beschäftigt sich mit der Verwendung von Modulen im Umfeld von Microservices und der Container-Technologien wie Docker.

Quellcodes des Buches

Alle Code-Beispiele und weitere Informationen finden sich unter: *http://Java-Modularisierung.de*

Für wen ist das Buch?

Entwickler

Das Buch richtet sich in erster Linie an Softwareentwickler, die die Modularisierung mit Java in eigenen Projekten nutzen wollen. Aber auch Softwarearchitekten können sich angesprochen fühlen, ist das Prinzip der Modularisierung doch auf einer höheren Abstraktionsstufe angesiedelt als die reine Erstellung von Klassen und Packages und erfordert bereits beim Entwurf eines Softwaresystems die entsprechende Berücksichtigung. Wer generell eine Einführung in das Thema Modularisierung sucht, wird hier ebenfalls fündig werden. Ebenso derjenige, der sich bereits mit Modularisierung auskennt, aber mehr über die Migration erfahren will oder wie sie im Kontext von z. B. Microservices und Containern Verwendung findet.

Architekten

Übersicht über die Kapitel

Teil I

Der erste Teil des Buches ist den Grundlagen zur Modularisierung gewidmet und den Zielen, die mit dem Java-Modularisierungskonzept verfolgt werden. Es wird ausführlich erläutert, was unter Modularisierung verstanden wird, was ein Modul ist und wie Module entworfen werden. Abgeschlossen wird der erste Teil mit der Betrachtung dessen, wie Modularisierung vor Java 9 gelöst wurde und welche Probleme mit dem Modularisierungskonzept gelöst werden.

Teil II

Teil II stellt ausführlich das Java-Modulkonzept vor. Es wird gezeigt, wie mit Java Module erstellt werden und wie eine modularisier-

te Anwendung gebaut wird. Zudem werden das modularisierte JDK und die sich daraus ergebenden neuen Möglichkeiten für Entwickler vorgestellt. Der Migration von Anwendungen hin zu Modulen ist ein weiteres Kapitel gewidmet. Zur Abrundung der Darstellung des Java-Modulsystems wird ein Vergleich zum OSGi-Modularisierungsframework gezogen. Abgeschlossen wird der zweite Teil mit konkreten Anleitungen, wie die gängigsten Entwicklungswerkzeuge bei der modularen Softwareentwicklung eingesetzt werden können, und ein »Real World«-Projekts zeigt, wie eine modulare Java-Anwendung gebaut wird.

Der letzte Teil des Buches wird den neuen Entwicklungen in der Welt der Softwareentwicklung gerecht. Nach dem Lesen dieses Teiles wird klar sein, was Microservices und Container-Technologien wie Docker mit der Modularisierung gemeinsam haben und wie Java-Module in Kombination mit diesen Technologien verwendet werden können.

Pfade durch das Buch

Um den Erwartungshaltungen der verschiedenen Lesern gerecht zu werden, gibt es auch verschiedene Möglichkeiten, sich den Inhalt des Buches zu erschließen. Während der eine möglichst schnell ohne allzu viel Theorie in die konkrete Anwendung von Modulen mit Java eingeführt werden will, möchte ein anderer vielleicht etwas mehr über die Konzepte dahinter erfahren. Ein weiterer Leser ist eventuell in OSGi sehr gut bewandert und interessiert sich für die Unterschiede und Gemeinsamkeiten zur Java-Modularisierung und was für Vorzüge ein modularisiertes JDK im Embedded-Bereich mit sich bringt. Für diese unterschiedlichen Anforderungen sind im Folgenden Leseempfehlungen bzgl. der Reihenfolge der Kapitel gegeben. Oft bietet es sich an, erst einmal das zu lesen, was am meisten interessiert, und sich danach erst mit dem Rest zu beschäftigen. Die einzelnen Kapitel sind weitestgehend autark und können leicht separat gelesen werden.

Fundierter Einstieg

- Einstieg in die Modularisierung
  Bei dieser Lesereihenfolge wird zunächst geklärt, was Modularisierung ist, wofür diese gut ist und was es mit dem modularisierten JDK auf sich hat. Danach werden die Modularisierung mit Java vorgestellt und die konkrete Verwendung.
  Kapitel 1 → 2.1 → 2.2 → 3 → 4

Schnelleinstieg

- Schnelleinstieg Module
  Wer schon weiß, was Module sind, und sofort wissen möchte, wie Modularisierung mit Java funktioniert, ist hier gut aufgehoben.
  Kapitel 3 → 4 → 10

- Für den OSGi-Kenner
  OSGi-Kenner
  Wer OSGi kennt, wird sich die Java-Modularisierung schnell erarbeiten können. Hilfreich dabei wird die Erkärung zu den Unterschieden und Gemeinsamkeiten beider sein.
  Kapitel 8 → 3 → 4

Danksagung

Mein Dank geht zunächst an meinen Lektor René Schönfeldt, der mich stets mit guten Tipps und Verbesserungsvorschlägen versorgt hat und an das gesamten dpunkt.team. Bei den mir unbekannten Gutachtern bedanke ich mich für viele Anregungen, die letztlich für die Zusammenstellung der in diesem Buch behandelten Themen mit gesorgt haben. Und zu guter Letzt bedanke ich mich bei meiner Frau Rosanne für ihre Geduld und Unterstützung, auch wenn es mal wieder sehr spät geworden ist.

Inhaltsverzeichnis

Teil I

Einführung und Grundlagen

Dieser Teil des Buches führt in das Thema Modularisierung ein. Zunächst wird das Prinzip der Modularisierung vorgestellt und gezeigt, warum Modularisierung so wichtig ist, was ein Modul ist und worauf bei dessen Entwurf zu achten ist. Zudem wird der Frage nachgegangen, warum überhaupt modularisiert werden sollte und was dies mit dem Bauen von langlebigen Architekturen zu tun hat. Abgerundet wird der einführende Teil mit einem kurzen Blick auf die Entstehungsgeschichte der Java-Module, welche Ziele mit der Einführung dieser verfolgt werden und wie bisher modularisiert wurde.

1 Das Prinzip der Modularisierung

Modularisierung ist ein grundlegendes Prinzip der Softwaretechnik. Sie hilft bei der Komplexitätsreduzierung und der Erhöhung der Flexibilität von Software und spielt eine wichtige Rolle bei der Wartbarkeit. Dieses Kapitel erläutert, was unter Modularisierung und dem damit verbundenen Modul-Begriff zu verstehen ist. Es wird gezeigt, wie Modularisierung angewendet wird und was dabei zu beachten ist. Dabei wird insbesondere das Zusammenspiel von Modulen untereinander betrachtet und beschrieben, was einen guten Modulentwurf kennzeichnet. Der Leser, der mit diesem Prinzip vertraut ist und für den Begriffe wie Kohäsion, Kopplung und Bindungen keine Fremdwörter sind, kann dieses Kapitel auch überspringen und mit Kapitel 3 fortfahren, wo von der Entstehung des Java-Modulsystems berichtet wird. Der Start mit dem Kernkapitel 3 ist ebenfalls möglich.

1.1 Was ist Modularisierung?

Softwaresysteme sind häufig große Systeme mit komplexem Charakter. Die Entwicklung solcher Systeme erfolgt in Teams, die zum Teil mehrere Jahre benötigen und Hunderttausende Zeilen Code produzieren oder noch viel mehr. Um diese Herausforderungen zu meistern, ist eine überlegte und strukturierte Vorgehensweise nötig.

Die Softwaretechnik (*engl. Software Engineering*) als Teilgebiet der Informatik definiert eine Reihe von Prinzipien und liefert Methoden, um Softwaresysteme jeglicher Größenordnung zu entwickeln. Im Kern geht es dabei fast immer um die Strukturierung komplexer Systeme mit dem Ziel, diese für den Entwickler intellektuell beherrschbar zu machen. Daher wundert es nicht, dass viele Prinzipien ihren Ursprung in der Art haben, wie wir Menschen versuchen, uns Komplexität begreifbar und damit beherrschbar zu machen. Eine zu lösende Aufgabe in immer kleinere Teilaufgaben zu gliedern ist dabei sicher einer der bekanntesten Vorgehensweisen.

Bereits im Römischen Reich stellte das Prinzip des Teilens und Herrschens (*lat. divide et impera*) ein wichtiges Paradigma dar. Spal-

te ein Volk in Untergruppen, damit diese leichter zu besiegen und zu beherrschen sind. Die moralische Beurteilung dieser politische Strategie der römische Außenpolitik außer Acht lassend, handelt es sich um eine durchaus kluge Herangehensweise.

Auf die Informatik angewandt bedeutet die römische Devise, teile das Problem in viele hinreichend kleine Teilprobleme und löse diese. Oder auf ein Softwaresystem bezogen: Zerlege das System in Teilsysteme. Für die Beherrschung dieser (Teil-)Systeme, wird die gleiche Vorgehensweise angewendet. Zerlege das (Teil-)System in weitere Teile. Und eben diese Zerlegung eines (Teil-)Systems wird als Modularisierung bezeichnet und die entstehenden Teile als Module.

Im Grunde ist auch die Unterteilung des Systems in Teilsysteme eine Art der Modularisierung. Eine Modularisierung im Großen, wenn man so will. Allerdings sei einschränkend erwähnt, dass eine Untergliederung in Teilsysteme nach unterschiedlichen Gesichtspunkten vorgenommen werden kann und dadurch bedingt ein Teilsystem nicht zwingend der Definition eines Moduls genügt. Daher kann nicht allgemein gesagt werden, dass auch ein Teilsystem ein Modul ist. Im Folgenden wird genau erläutert, was ein Modul ist und welche Kriterien dieses erfüllen muss.

Modularisierung

Definition: Modularisierung

Modularisierung bedeutet die Zerlegung eines Systems in Module, unter Berücksichtigung bestimmter Kriterien.

Das Prinzip der Modularisierung findet sich in vielerlei Bereichen wieder; beispielsweise im Bereich der Elektrotechnik. Ein *herkömmlicher* Computer ist aus verschiedenen Baugruppen aufgebaut. Jedes dieser Teile stellt dabei ein austauschbares Modul dar. Zum Beispiel kann die Grafikkarte bei einem Defekt leicht ausgebaut und durch eine neue Karte ersetzt werden. Das Prinzip der Modularisierung lässt sich auch weiter fassen. Am Computer sind ein Drucker und ein Monitor angeschlossen, die sich ebenfalls ersetzen lassen. Oder der heimische Fernseher, der eine Verbindung zu einer Spielekonsole und einem DVD-Player hat, die wiederum mit einem Verstärker verbunden sind. In solch einem Szenario können neue Geräte leicht hinzugefügt oder entfernt werden.

Die Eigenschaft der Austauschbarkeit und das leichte Hinzufügen und Entfernen neuer Geräte deutet schon darauf hin, dass die Verbindungsstellen nicht beliebig sein dürfen. Definierte Schnittstellen sind die Grundvoraussetzung für die Austauschbarkeit, das Hinzufügen und Entfernen von Baugruppen und für die Kommunikation der Module

untereinander. Im Bereich der Elektrotechnik sind dies beispielsweise USB-Ports und HDMI-Anschlüsse und die Art der Signalverarbeitung.

Ein weiteres Beispiel für eine modulare Bauweise im größeren Maßstab findet sich beim Blick in unseren Erdorbit. Als größtes künstliches Objekt zieht dort die internationale Raumstation ISS ihre Kreise um unseren Planeten. Die ISS als Projekt von fünf Raumfahrtagenturen und weiteren zehn Ländern besteht aus 34 Modulen, die in den nächsten Jahren noch um sechs weitere ergänzt werden soll. Die Module sind in unterschiedlichen Ländern von unterschiedlichen Teams gebaut worden und wurden nach erfolgreichen Tests auf dem Boden mit Trägerraketen und Raumfähren in die Erdumlaufbahn gebracht, um dort an die Station angedockt zu werden. Das Innenleben der Module ist unabhängig von den anderen Modulen gebaut worden und auf die gewünschte Funktionalität hin testbar. Auch hier muss beim Bau beachtet werden, an welche anderen Module diese gekoppelt werden, ob Funktionen anderer Module genutzt werden sollen oder welche eigenen Funktionalitäten anderen Modulen zu Verfügung zu stellen sind und wie die Verbindungsstellen zwischen den Modulen aussehen müssen.

Die Beispiele zeigen, dass Modularisierung grundsätzlich dabei hilft, durch die Aufteilung eines Systems in Module, die Komplexität zu verringern, da die einzelnen Module getrennt voneinander betrachtet und verstanden werden können. Dies wiederum unterstützt die Wartbarkeit der einzelnen Module. Darüber hinaus vereinfachen die von der Modularität geforderten definierten Schnittstellen zwischen den Modulen die Erweiterbarkeit des Systems. Und die Rekombination von Modulen erlaubt die Erstellung von verschiedenen Varianten des Systems. Letzteres ist Grundlage der Produktlinienentwicklung (*engl. Product Line Engineering*), welches auch in Bereichen außerhalb der Softwarewelt zu finden ist. Zum Beispiel ist die Automobilindustrie bemüht, die einzelnen Automobilvarianten nur noch auf Basis einer einheitlichen Plattform, also in Form von Modulen, zusammenzubauen. Abbildung 1–1 zeigt, dass früher jedes Automodell eine eigene Entwicklungslinie war, heutzutage aber immer mehr eine modulare Bauweise angewendet wird. So kommt z. B. ein Motorblock in verschiedenen Automodellen zum Einsatz mit dem zukünftigen Ziel, verschiedene Modelle aus einer Menge an Modulen zusammenzubauen.

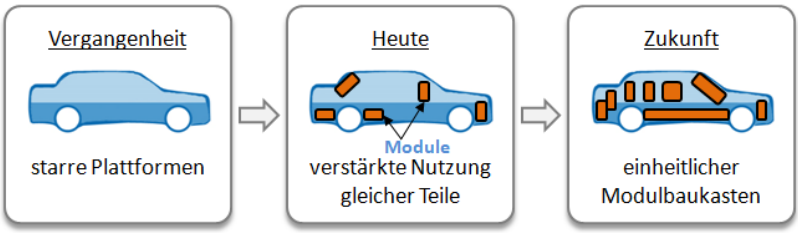


Abb. 1–1
Modularer Autobau

Ziele der Modularisierung

Ziele der Modularisierung

- Beherrschbarkeit von Komplexität
- bessere Erweiterbarkeit und Wartbarkeit
- bessere Verständlichkeit
- größere Wiederverwendbarkeit
- Schaffung neuer Systeme durch Rekombination von Modulen

Modularisierung bringt eine Reihe weiterer Vorteile mit sich, die den Softwareentwicklungsprozess direkt oder indirekt positiv beeinflusst. Beispielsweise können Module von verschiedenen Entwicklerteams unabhängig und parallel entwickelt werden, um den Herstellungsprozess des Gesamtsystems zu beschleunigen.

Um das Prinzip der Modularisierung konkret anwenden und ein System sinnvoll zerlegen zu können, ist eine genauere Betrachtung dessen, was ein Modul ist, notwendig. Der folgende Abschnitt geht dieser Frage nach und liefert das nötige Fundament für das spätere Verständnis von Java-Modulen.

1.2 Was ist ein Modul?

Die zuvor aufgeführten Ziele der Modularisierung liefern bereits eine Idee davon, was für Anforderungen Module erfüllen müssen, um von einem Modul sprechen zu können. Zunächst einmal erfüllt ein Modul einen abgeschlossenen Aufgabenbereich und beinhaltet die dafür nötigen Operationen und Daten.

Operationen eines Moduls

Definition: Operationen eines Moduls

Operationen spezifizieren, was ein Modul anderen Teilen des Systems an Funktionalitäten zur Verfügung stellt. Operationen sind ausführbare Methoden.

Die Kommunikation eines Moduls mit der Außenwelt, also der Zugriff auf das Modul und der Zugriff von diesem Modul auf andere Module, erfolgt über eindeutig spezifizierte Schnittstellen. Abbildung 1–2 zeigt den grundsätzlichen Aufbau.

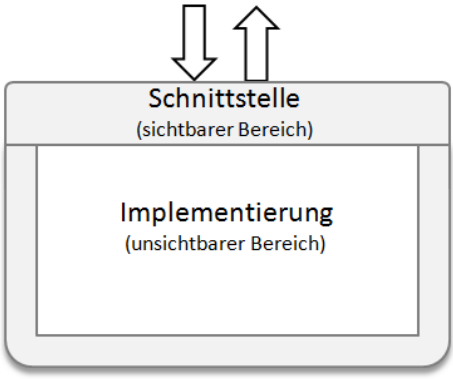


Abb. 1–2
Grundsätzlicher Aufbau eines Moduls

Das Modul fungiert als eine Art Behälter für Objekte, der aus einem unsichtbaren und einem sichtbaren Teil besteht. Der sichtbare Teil ist die Schnittstelle des Moduls und ist die Aufzählung derer Objekte (Datenstrukturen, Datentypen, Methoden, Module), die das Modul nach außen hin zur Verfügung stellt. Der Zugriff auf diese erfolgt über definierte Operationen in der Modulschnittstelle. Der unsichtbare Teil beherbergt die eigentliche Implementierung, also die ausimplementierten Operationen und Daten. Mit Abbildung 1–3 wird ein genauerer Blick auf den Aufbau eines Modules und die Verbindung zu anderen Modulen geworfen.

Zu sehen sind die drei Module A, B und C. Modul A stellt über eine Schnittstelle seine Operationen der Umgebung zur Verfügung. Hierbei wird von der Exportschnittstelle gesprochen. Innerhalb des Moduls

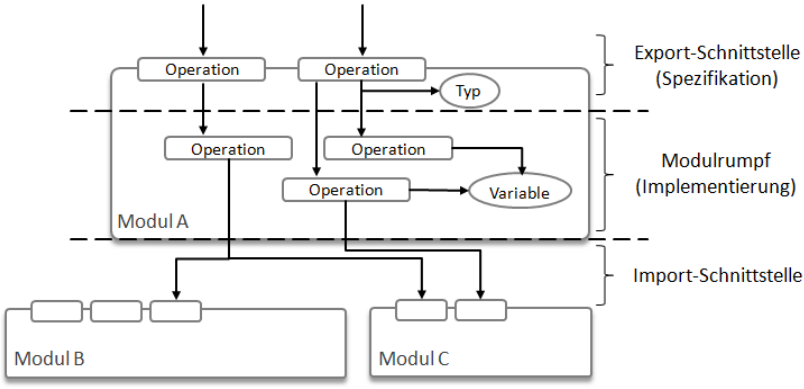


Abb. 1–3
Schematischer Aufbau eines Moduls

befinden sich die Implementierungen dieser Operationen. Wie auf der Abbildung zu sehen, benötigen einige der Operationen von Modul A, Operationen von den Modulen B und C. Um diese Operationen der anderen Module zu verwenden, müssen diese gezielt angefordert werden. Die entsprechende Referenzierung auf die Schnittstellen von Modul B und C erfolgt innerhalb von Modul A und wird als Importschnittstelle bezeichnet. Wenn im folgenden Text nur von Schnittstelle des Moduls die Rede ist, dann ist immer die Exportschnittstelle gemeint.

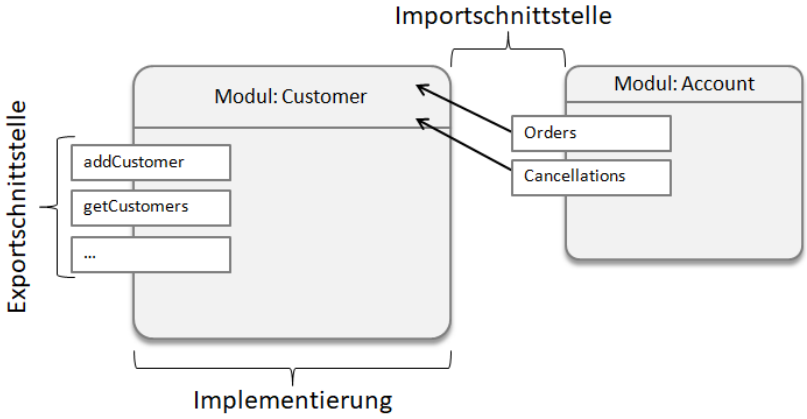
Export- und Imporstschnittstellen eines Moduls

Definition: Export- und Importschnittstellen eines Moduls

- Die Exportschnittstelle gibt an, welche Operationen und Daten anderen Modulen zur Verfügung gestellt werden.
- Die Importschnittstelle gibt an, welche Operationen und Daten ein Modul von anderen Modulen benötigt.

Um ein konkretes Beispiel zu wählen, zeigt Abbildung 1–4 die beiden Module Customer (*dt . Kunde*) und Account (*dt. Konto*) eines fiktiven Web-Shops. Das Modul Customer enthält alle kundenspezifischen Operationen und stellt unter anderem die beiden Operationen `addCustomer` und `getCustomers` über die Exportschnittstelle für die Umgebung zur Verfügung. Das Modul selber benötigt von dem anderen Modul Account, die beiden Objekte `Orders` (*dt. Bestellungen*) und `Cancellations` (*dt. Stornierungen*). Dieses Modul ist für alle Dinge rund um den Warenkorb zuständig.

Abb. 1–4
Beispiel Module



Der Pseudocode zur Modul-Spezifikation von Kunde könnte so aussehen:

Listing 1–1
Modul-Spezifikation von Kunde

```
DEFINITION MODUL Customer {
        FROM Account IMPORT Orders, Cancellations;
        METHOD addCustomer(User user)
        METHOD getCustomers(List users)
        ...
}
```

Die Implementierung der Operationen eines Moduls können verändert werden, ohne dass die Schnittstelle zwingend angepasst werden muss. Zudem haben von außen zugreifende Module keine Kenntnis vom inneren Arbeiten des Moduls und benötigen diese auch nicht. Das Verbergen der Implementierungsdetails entspricht dem Geheimnisprinzip. Der Zugriff auf die Implementierung über Schnittstellen wird als Prinzip der Kapselung bezeichnet. Beides sind wesentliche Entwurfsprinzipien für die Erstellung modularer Systeme und werden daher im Folgenden genauer betrachtet.

Modul

Definition: Modul

- Zusammenfassung von Operationen und Daten zur Realisierung einer in sich abgeschlossenen Aufgabe
- Kommunikation mit der Außenwelt nur über eine eindeutig spezifizierte Schnittstelle
- Nutzung des Moduls möglich ohne Kenntnis des inneren Ablaufs
- Korrektheit des Moduls durch Tests nachprüfbar ohne Kenntnis seiner Einbettung

  (Anmerkung: In der Praxis werden Module meist zu mockende Abhängigkeiten besitzen.)

1.2.1 Geheimnisprinzip und Datenkapselung

Das Geheimnisprinzip (*engl. Information Hiding*) geht auf eine Arbeit von David L. Parnas im Jahre 1972 zurück [20]. Es bezeichnet ein Entwurfsprinzip für Module. Häufig wird das Geheimnisprinzip mit der Datenkapselung gleichgesetzt, was aber nicht ganz korrekt ist. Beim Geheimnisprinzip geht es rein um die Sichtbarkeit von Objekten. Es geht nicht um die Zugriffsrechte der außerhalb liegenden Objekte auf die nicht sichtbaren Objekte. Datenkapselung hingegen ist eine Verschärfung des Geheimnisprinzips und betrifft die Festlegung von Zu-

griffsrechten von äußeren Objekten. In diesem Sinne ist Datenkapselung eine Form von Information Hiding.

Dabei wird die Implementierung einer Datenstruktur von ihren sichtbaren Eigenschaften getrennt. Das Ergebnis wird als abstrakte Datenstruktur (also die Implementierung eines ADT (*Abstrakter Datentyp*)) bezeichnet, über welche dann der Zugriff erfolgt. Die eigentliche Datenstruktur ist nicht sichtbar. Konkret bedeutet das nichts anderes, als dass über eine Schnittstelle zugegriffen wird. Diese Schnittstelle besteht aus Operationen, die den Umgang mit der Datenstruktur beschreiben.

Auf Module bezogen bedeutet das Prinzip der Kapselung, dass das Modul als eine Art Blackbox betrachtet werden kann, die nur relevante Informationen nach außen gibt. Die Implementierung des Moduls hingegen bleibt hinter der Schnittstelle verborgen. Der Zugriff erfolgt ausschließlich über Zugriffsmethoden, die über die Schnittstelle zur Verfügung gestellt werden.

Das Geheimnisprinzip wird im Wesentlichen in drei Schritten angewandt:

1. Alle zusammenhängenden und veränderbaren Teile finden, die sich im Leben des Softwaresystems ändern könnten. In erster Linie handelt es sich um die Daten selber oder generell um Bereiche der Software, die sich vermutlich ändern werden.
2. Das System in Module aufspalten, wobei jedes Modul genau eine dieser Teile kapselt. Die so gekapselten Teile werden als Geheimnis bezeichnet.
3. Die Schnittstellen der Module entwerfen. Dabei sollten sich die Schnittstellen nicht ändern müssen, selbst wenn das Geheimnis geändert würde.

Die sich daraus ergebenden Vorteile sind vielfältig. Änderungen wirken sich nicht so stark auf das gesamte Softwaresystem aus, da diese nur innerhalb eines Moduls stattfinden, unabhängig von anderen. Oder die Änderungen beschränken sich zumindest auf ein paar wenige Module. Die Wiederverwendbarkeit, falls gewünscht, wird erhöht, liegen die Geheimnisse doch gekapselt vor. Die Verbindungen zwischen den Modulen sind zudem geregelt und übersichtlich.

Geheimnisprinzip (*engl. Information Hiding*)

Geheimnisprinzip (Information Hiding)

Bezeichnet nach Parnas ein Entwurfsprinzip für Module.

- Module verbergen Teile eines Softwaresystems, die als zusammenhängend und potenziell veränderbar identifiziert wurden.
- Die Modul-Details bleiben nach außen hin verborgen und nur für den Aufrufer relevante Informationen werden gezeigt.

Für die Modularisierung eines Systems, müssen neben der Kenntnis davon, wie ein Modul aufgebaut ist, noch weitere Fragen geklärt werden. Zum Beispiel, in wie viele logische Teile, also Module ein System aufgeteilt werden soll? Wie groß dürfen die Schnittstellen sein? Gibt es Probleme beim Aufbau einer Modulhierarchie? Diese und weitere Fragen ergeben sich zwangsläufig bei der Modularisierung und sind Gegenstand des nächsten Abschnitts.

1.3 Modularisierung eines Systems

Die eigentliche Kunst der Modularisierung liegt in der Aufteilung des Systems und der Definition der Schnittstellen, über welche die Module miteinander kommunizieren. Dabei muss zunächst die Frage beantwortet werden, welche Teile eines Softwaresystems zu Modulen zusammengefasst werden sollen. Danach rückt das Zusammenspiel der Module untereinander in den Fokus. Nachfolgend wird zunächst die grundsätzliche Vorgehensweise erläutert und danach werden Kriterien für den Entwurf vorgestellt, die Anhaltspunkte dafür liefern, wie eine gute Zerlegung in Module erarbeitet werden kann.

1.3.1 Entwurfsprozess für Module

Der in diesem Abschnitt beschriebene Entwurfsprozess von Modulen ist nur ein Teil des Gesamtprozesses für die Erstellung einer Architektur. Abbildung 1–5 zeigt den grundsätzlichen Ablauf beim Entwurf einer modularen Struktur.

1. Zerlegung in Module
2. Überprüfung der Kriterien (s. Kapitel 1.3.3)
3. Revision und Verfeinerung

Im ersten Schritt wird ein konkretes Entwurfsproblem in kleinere Teilprobleme zerlegt. Ziel ist die Komplexitätsreduzierung von einem komplexen Ausgangsproblem zu weniger komplexen Teilproblemen. Die

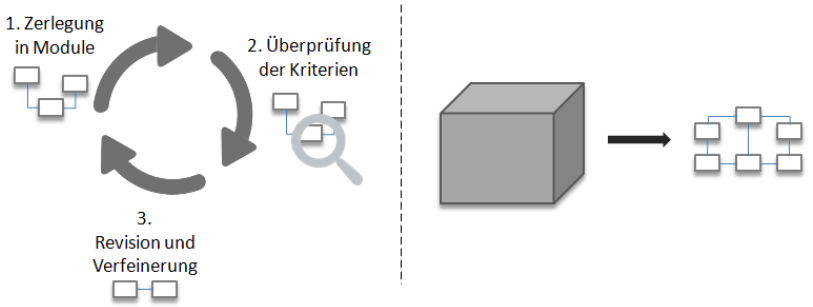


Abb. 1–5 *Zerlegung des Entwurfsproblems in einem zyklischen Prozess*

Teilprobleme werden als weitestgehend unabhängige Module abgebildet, die über eine möglichst einfache Struktur miteinander verbunden sind. In der Praxis erfordert dieser erste Schritt je nach System schon einige Erfahrung. Nicht jedes Problem lässt sich so leicht in weniger komplexe Teilprobleme aufteilen. Häufig wird sich einer mehrstufigen Zerlegung beholfen, in der zu große Module als Teilsysteme aufgefasst werden, die dann wiederum modularisiert werden. Die Schnittstelle eines solchen Teilsystems ist dann die Summe der Schnittstellen von den im Teilsystem enthaltenen Modulen. Wo die Module geschnitten werden, hängt immer vom konkreten Problem ab und ist zuletzt immer eine individuelle Angelegenheit, wo auch die Kreativität des Architekten gefragt ist.

Anhand von bestimmten Kriterien kann der Qualitätsgrad der bis dahin vorgenommenen Modularisierung überprüft werden. Diese Kriterien finden natürlich schon im ersten Schritt ihre Anwendung. Zudem müssen die Kriterien nicht sklavisch genau befolgt werden. Eine gute Architektur ist nie das alleinige Ergebnis von Regeln, die befolgt wurden.

Der dritte Schritt ergibt sich als Ergebnis aus dem vorangegangenen Schritt. Hier wird die Modularisierung verfeinert und korrigiert.

Die in den nächsten Abschnitten behandelten Entwurfstechniken und Entwurfskriterien helfen dabei, eine Zerlegung in Module vorzunehmen.

1.3.2 Entwurfstechniken

Die Aufteilung eines Entwurfsproblems in kleinere Teilprobleme ist kein trivialer Vorgang. Entwurfstechniken helfen hier bei der Aufteilung und der Zuordnung der gefundenen Teilprobleme zu Modulen. Das über alles thronende Prinzip ist das Prinzip der Abstraktion. Jede Aufteilung in Module ist immer das Ergebnis von Abstraktion, die als eine Basisqualifikation menschlicher Kognition in einer starken Aus-

prägung zugleich eine Schlüsselqualifikation des Softwarearchitekten ist. Bei der Abstraktion wird das Wesentliche vom Unwesentlichen getrennt, wodurch Ordnungen, Klassifizierungen und Einteilungen vorgenommen werden. Das Gegenstück der Abstraktion ist die Konkretisierung und die wechselnde Anwendung beider ist der Schlüssel für jede schrittweise Zerlegung eines Systems in Module.

Entwurfstechniken können nach verschiedenen Gesichtspunkten eingeteilt werden. Eine Einteilung nach der Richtung der Abstraktion zeigt Abbildung 1–6. Als Ausgangspunkt kann die allgemeine Lösung bzw. das Ergebnis, wie es sich für den Anwender der Software darstellt, genommen werden und davon ausgehend werden die Anforderungen abgeleitet. Diese Entwurfsrichtung wird als Top-down bezeichnet. Ein anderer Ausgangspunkt ist die Betrachtung einer konkreten Anforderung hin zur allgemeinen Lösung des Systems.

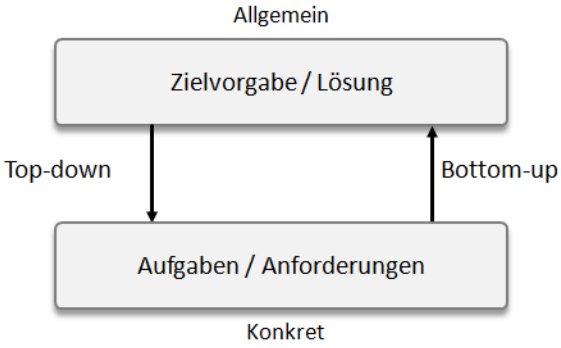


Abb. 1–6
Entwurfsrichtungen

Die nächsten Abschnitte beleuchten diese Vorgehensweisen im Kontext der Modularisierung genauer. Dabei geht es nur um die Darstellung der grundsätzlichen Vorgehensweisen und weniger um die Bewertung dieser oder eine Gegenüberstellung mit anderen Möglichkeiten, die den Rahmen der Kapitel sprengen würde. Beispielsweise ließe sich der Versuch eines umfassenden Lösungsentwurfs vor der Realisierung eines Systems (Big Design Upfront (BDT)) mit den folgenden Methoden kontrovers diskutieren. Verursacht dies zwar höhere Aufwände vor der Realisierung, birgt es aber auch das Potenzial, spätere Anpassungen durch Antizipation zu vermeiden, was eine Kosten-Nutzen-Abwägung im Vorfeld durchaus betrachtenswert macht.

Top-down

Top-down-Entwurf

Bei der Top-down-Vorgehensweise liegt zu Beginn eine abstrakte Zielvorgabe für die Funktionalität des Systems vor. Mit abstrakter Zielvorgabe ist eine allgemeine Lösungsidee gemeint, die noch keinerlei Details

für die Realisierung enthält. Die Vorgehensweise sieht nun so aus, dass die Zielvorgabe in Teilaufgaben zerlegt wird. Dabei handelt es sich um eine schrittweise Verfeinerung, wo die gefundenen Teilaufgaben so lange heruntergebrochen werden, bis ein für ein Modul geeignete Teilaufgabe entsteht. Hierbei wird auch von der Dekomposition des Systems gesprochen.

Eine Gesamtsicht als Ausgangslage bietet allgemein eine gute Möglichkeit zur Strukturierung und führt im Ergebnis genau zur gewünschten Zielvorgabe. Eine Herausforderung dabei ist, dass konkrete Realisierungsprobleme häufig erst später bekannt werden und die entstandenen Teillösungen bzw. Module häufig nicht unter dem Aspekt der Wiederverwendbarkeit betrachtet werden.

Die ausschließliche Anwendung dieser Vorgehensweise führt allerdings nicht unbedingt zu einer guten Modularisierung, sind Funktionen innerhalb von Modulen doch oft kreuz und quer in der Systemhierarchie verteilt.

Bottom-up

Bottom-up-Entwurf

Bei der Bottom-up-Vorgehensweise wird von konkreten Bausteinen ausgehend schrittweise das System zusammengestellt. Hierbei wird auch von der Komposition des Systems gesprochen. Es findet eine schrittweise Zusammensetzung von Bausteinen zu immer größeren Einheiten statt. Also konkrete in Klassen gegossene Teilaufgaben werden zu Modulen zusammengefasst, die wiederum zu größeren Einheiten zusammengesetzt das System ergeben.

Ein großer Vorteil bei dieser Vorgehensweise ist, dass Realisierungsprobleme frühzeitig auffallen und Teilaufgaben schnell gefunden werden. Für die Modularisierung ist diese Vorgehensweise sehr hilfreich, aber führt auch hier nicht unbedingt zum idealen Ergebnis. Die Herausforderung liegt hier darin, die Gesamtsicht im Auge zu behalten und das Ziel zu erreichen. Zudem wird die Entwicklung von Komponenten begünstigt, die in Anbetracht der späteren Gesamtschau für das System gar nicht nötig waren oder gar nicht mehr benötigt werden.

Up-And-Down

Kombination Top-down- und Bottom-up-Entwurf

Ein guter Entwurf ist meist die Summe von Erfahrung, Abstraktionsvermögen und Techniken. Bezüglich der beiden grundsätzlich Vorgehensweisen Top-down und Bottom-up hat sich neben anderen Techniken eine Kombination dieser beiden in der Praxis bewährt. Von erfahrenen Entwicklern und Architekten wird sie meist schon intuitiv angewendet.

Als Ausgangslage werden die Zielvorgabe des Systems und mögliche Lösungsbausteine für konkrete Funktionalitäten betrachtet. Hierbei

wird die Zielvorgabe in Teilaufgaben zerlegt und diese werden durch die Bausteine konkretisiert. Was an dieser Stelle etwas abstrakt wirkt, sieht im objektorientierten Ansatz in etwa so aus, dass für einen konkreten Baustein, für eine konkrete Aufgabe, die benötigten Objekte identifiziert und für die daraus resultierenden Klassen, die erforderlichen Methoden ermittelt werden. Mit den entstehenden Klassen können wiederum größere Einheiten wie z. B. Module erstellt werden, die die Teilaufgabe erfüllen. Diese Technik unterstützt die Erstellung von Modulen und den Entwurf einer möglichst ökonomischen Lösung unter der Berücksichtigung der Zielvorgabe. Die Herausforderung hier ist die wechselseitige Bearbeitungsrichtung, die eine strukturierte Vorgehensweise beeinträchtigen kann.

Das folgende Kapitel beschäftigt sich mit der Beurteilung der Modularisierungsqualität eines Systems und den Entwurfskriterien zur Erstellung von Modulen.

1.3.3 Entwurfskriterien zur Modularisierung

Die Softwaretechnik liefert eine Reihe von Kriterien, die zur Beurteilung der Modularisierungsqualität eines Systems herangezogen werden können und dabei helfen, Module zu entwerfen und deren Zusammenspiel zu berücksichtigen. Dabei stellen diese Kriterien Empfehlungen und Richtwerte dar und sind je nach System unterschiedlich zu gewichten. In den folgenden Abschnitten werden diese Kriterien kurz vorgestellt und erläutert.

Entwurfskriterien zur Modularisierung

Entwurfskriterien zur Modularisierung eines Systems

Bei der Modularisierung sind folgende Entwurfskriterien zu berücksichtigen:

- Modulgeschlossenheit
- Maximale Modulbindung
- Minimale Modulkopplung
- Minimale Schnittstelle
- Modulanzahl
- Modulgröße
- Testbarkeit
- Seiteneffektfreiheit
- Importzahl
- Modulhierarchie

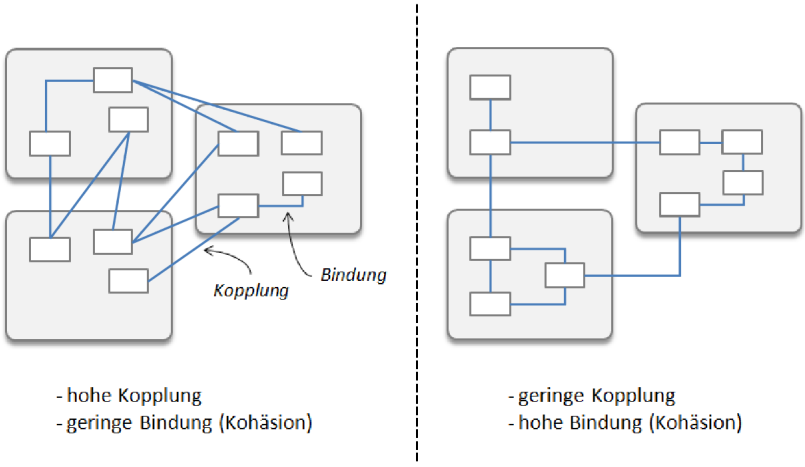
Modulgeschlossenheit

Modulgeschlossenheit bedeutet, dass ein Modul für eine in sich geschlossene Aufgabe zuständig ist. Durch die Forderung nach Modulgeschlossenheit, wird erreicht, dass Module ausgetauscht werden können und dadurch die Flexibilität und die Erweiterbarkeit des Systems erhöht werden. Zudem führen Änderungen an der Aufgabe nur zu Änderungen in dem Modul und lassen andere Module außen vor.

Modulbindung und Modulkopplung

Das Hauptziel der Modularisierung ist die Verminderung von Komplexität. Wenn allerdings die Anzahl der Module im Verhältnis zur Gesamtgröße des Systems in keinem vernünftigen Verhältnis steht, kann dies wiederum die Komplexität erhöhen. In diesem Fall würden sehr viele Schnittstellen existieren und es gäbe viele Abhängigkeiten zwischen den Modulen. Unter diesem Gesichtspunkt kann Modularisierung eine echte Herausforderung sein. Bei der im Folgenden vorgestellten Modulbindung (Kohäsion) und Modulkopplung geht es generell um Abhängigkeiten. Die Kohäsion handelt von Abhängigkeiten innerhalb eines Moduls und die Kopplung von Abhängigkeiten zwischen Modulen. Abbildung 1–7 stellt den Zusammenhang zwischen der Modulkopplung und Modulbindung grafisch dar.

Abb. 1–7
Zusammenhang Kopplung und Bindung



Maximale Modulbindung

Modulbindung (Kohäsion) Die Modulbindung ist der Grad für den inneren Zusammenhang eines Moduls. Dieser sollte möglichst groß sein. Groß bedeutet, dass alle logisch zusammengehörigen Operationen und Daten für die Lösung einer Aufgabe in einem Modul zusammen-