Ein Buch zum Mitmachen und Verstehen

Objektorientierte Analyse & Design

von Kopf bis Fuß

Verbessern Sie Ihre Kommunikationsfähigkeit mit UML und Anwendungsfällen

Trainieren Sie Ihren Kopf mit Dutzenden von OO-Übungen

Vermeiden Sie unzufriedene Kunden

Machen Sie aus Ihren Anforderungen und Entwürfen richtige Software

Laden Sie sich die entscheidenden OO-Entwurfsprinzipien direkt ins Hirn

Erfahren Sie, wie Tosca mit Hilfe von Abstraktion, Aggregation und Delegation Objekthausen umfahren konnte

Brett D. McLaughlin, Gary Pollice & David West
Deutsche Übersetzung von Lars Schulten

O'REILLY®

Objektorientierte Analyse und Design
von Kopf bis Fuß

Brett D. McLaughlin
Gary Pollice
David West

Deutsche Übersetzung von
Lars Schulten

O'REILLY®

Beijing • Cambridge • Köln • Paris • Sebastopol • Taipei • Tokyo

Kommentare und Fragen können Sie gerne an uns richten:
O'Reilly Verlag
Balthasarstr. 81
50670 Köln
Tel.: 0221/9731600
Fax: 0221/9731608
E-Mail: kommentar@oreilly.de

Die Originalausgabe erschien 2006 unter dem Titel
Head First Object-Oriented Analysis & Design bei O'Reilly Media, Inc.

Bibliografische Information Der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über *http://dnb.ddb.de* abrufbar.

Übersetzung und deutsche Bearbeitung: Lars Schulten, Köln
Lektorat: Christine Haite, Köln
Fachgutachten: Karsten Loesing, Bamberg & Anke Werner, Dortmund
Korrektorat: Sibylle Feldmann, Düsseldorf
Satz: Conrad Neumann, München
Umschlaggestaltung: Mike Kohnke & Edie Freedman, Boston
Produktion: Andrea Miß, Köln
Belichtung, Druck und buchbinderische Verarbeitung: Media-Print, Paderborn

ISBN-10 3-89721-495-4
ISBN-13 978-3-89721-495-8

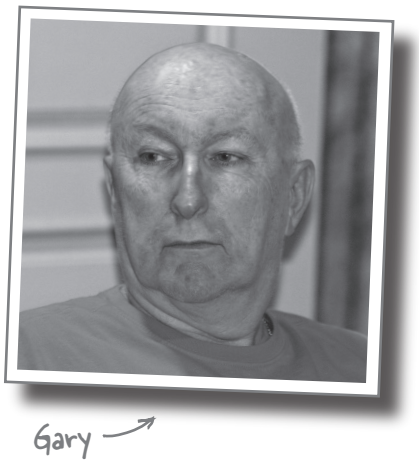
Dieses Buch ist auf 100% chlorfrei gebleichtem Papier gedruckt.

All den brillanten Menschen, denen die unterschiedlichen Möglichkeiten eingefallen sind, Anforderungen zu sammeln, Software zu analysieren und Code zu entwerfen ...

... danke, dass ihr etwas ausgedacht habt, das gut genug ist, um gute Software zu entwickeln, und trotzdem so schwer, dass wir dieses Buch brauchten, um es zu erklären.

Brett McLaughlin ist ein Gitarrenspieler, der immer noch mit der Einsicht kämpft, dass man seine Rechnungen nicht zahlen kann, wenn man akustischen Fingerstyle-Blues und -Jazz spielt. Aber vor nicht allzu langer Zeit hat er zu seiner Freude entdeckt, dass das Schreiben von Büchern, die anderen Menschen helfen, bessere Programmierer zu werden, dazu beiträgt, Rechnungen zu bezahlen. Darüber ist er sehr glücklich, genau wie seine Frau Leigh und seine Jungs Dean und Robbie.

Bevor Brett ins Von Kopf bis Fuß-Land gewandert ist, hat er für Nextel Communications und Allegiance Telecom Java Enterprise-Anwendungen aufgebaut. Als ihm das zu banal wurde, hat Brett sich Application-Servern zugewandt und an den Interna der Lutris Enhydra Servlet-Engine und EJB-Containers gearbeitet. Auf diesem Weg wurde Brett süchtig nach Open Source-Software und war an der Gründung mehrerer cooler Programmierwerkzeuge wie Jakarta Turbine und JDOM beteiligt. Schreiben Sie ihm an *brett@oreilly.com*.



Gary Pollice bezeichnet sich selbst als Griesgram (das ist ein mürrischer, übellauniger, in der Regel alter Mann), der mehr als 35 Jahre in der Industrie verbracht hat, um herauszufinden, was er werden will, wenn er erwachsen geworden ist. Auch wenn er noch nicht erwachsen geworden ist, hat er 2003 den Sprung in die heiligen Hallen der Akademie gemacht, wo er die Köpfe der nächsten Generation von Softwareentwicklern mit radikalen Ideen wie diesen verdirbt: »Entwickeln Sie Software für Ihren Kunden. Lernen Sie, als Teil eines Teams zu arbeiten. Design und Codequalität und Eleganz und Richtigkeit zählen. Es ist okay, wenn man ein Nerd ist, solange man ein guter Nerd ist.«

Gary ist Professor of Practice (was bedeutet, dass er einen richtigen Job hatte, bevor er Professor wurde) am Worcester Polytechnic Institute. Er lebt in Zentral-Massachusetts mit seiner Frau Vikki und ihren beiden Hunden Aloysius und Ignatius. Sie können seine WPI-Homepage unter **`http://web.cs.wpi.edu/~gpollice/`** besuchen. Sie können ihm gern Anmerkungen zukommen lassen oder sich über das Buch beschweren – oder es loben.

Dave West würde sich selbst als Sheik-Geek bezeichnen. Unglücklicherweise würde ihn sonst keiner so beschreiben. Sie würden sagen, dass er ein professioneller Engländer sei, der am liebsten mit der Leidenschaft und Energie eines Laienpredigers über bewährte Verfahren zur Softwareentwicklung spricht. Vor Kurzem hat Dave zu Ivar Jacobson Consulting gewechselt, wo er für Nord-, Süd- und Mittelamerika verantwortlich ist und seine Leidenschaften kombinieren kann, über Softwareentwicklung zu reden, über Rugby und Football zu diskutieren und zu behaupten, dass Kricket aufregender sei als Baseball.

Bevor er für Ivar Jacobson Consulting die Amerikas übernommen hat, arbeitete Dave eine Reihe von Jahren bei Rational Software (jetzt ein Teil von IBM). Bei Rational und IBM nahm Dave viele Positionen ein, unter anderem als Product Manager für RUP, wo er die Konzepte von Prozess-Plugins und Agilität in RUP eingeführte. Sie können ihn unter *dwest@ivarjacobson.com* kontaktieren.

Über den Übersetzer dieses Buchs (der bei aller Berühmtheit nicht auf der Straße erkannt werden will)

Lars Schulten ist freier Übersetzer für IT-Fachliteratur und hat für den O'Reilly Verlag schon unzählige Bücher zu ungefähr allem übersetzt, was man mit Computern so anstellen kann. Eigentlich hat er mal Philosopie studiert, aber mit Computern schlägt er sich schon seit den Zeiten herum, in denen Windows laufen lernte. Die Liste der Dinge, mit denen er sich beschäftigt, ist ungefähr so lang, launenhaft und heterogen wie die seiner Lieblingsessen oder Lieblingsbücher.

Lars legt sich eben nicht gern fest. »Eine Klasse, eine Verantwortlichkeit«, das ist sicher nicht seine Sache. Am besten funktioniert er, wenn man ihn als Universaladapter betrachtet und verdachtsweise einfach mal eine Methode aufruft und sich dann vom Ergebnis überraschen lässt.

Allein tritt er eigentlich nur auf, wenn er mal wieder versucht, den körperlichen Verfall mit sportlicher Betätigung aufzuhalten. Sonst ist er immer in Begleitung eines Buchs, seines Laptops oder Frederics unterwegs. Frederic ist vier Jahre alt und setzt gerne eine sehr kritische Miene auf, wenn Papa die Spielerei mit dem Conpuuta als Arbeit bezeichnet.

Zur deutschen Übersetzung

Das leidige Thema der deutschen Umlaute im Code haben wir natürlich auch gründlich diskutiert. Der besseren Lesbarkeit wegen haben wir schließlich in allen Arten von Bezeichnern Umlaute verwendet – auch in Klassennamen. Solange Sie innerhalb Ihres Systems bleiben und die Klassen nur dort kompilieren und ausführen, funktioniert dies in der Regel. Aber sobald Sie JARs erzeugen und/oder Ihre Programme auf anderen Plattformen einsetzen möchten, könnte es zu Problemen kommen. Sie sollten dann unbedingt in allen Verzeichnis- und Klassennamen (auch bei inneren Klassen!) auf Umlaute, auf das »ß« und Ähnliches verzichten. Profis, die meist sowieso zu englischen Bezeichnern neigen, empfinden Umlaute oft als irritierend und praxisfern – unser Anliegen ist es aber, dem lernenden Leser den Text, auch den Code, so leicht verdaulich zu präsentieren wie möglich.

Dies ist ein Auszug aus dem Buch *„Objektorientierte Analyse und Design von Kopf bis Fuß“*, ISBN 978-3-89721-495-8
http://www.oreilly.de/catalog/hfobjectsger/

Weitere verwandte Bücher von O'Reilly

Die Kunst des IT-Projektmanagements

UML 2.0 in a Nutshell

Practical Development Environments

Process Improvement Essentials

Prefactoring

Ajax Design Patterns

Learning UML

Applied Software Project Management

Unit Test Frameworks

Weitere Bücher in O'Reillys *Von Kopf bis Fuß*-Reihe

Entwurfsmuster von Kopf bis Fuß

Java von Kopf bis Fuß

Ajax von Kopf bis Fuß

HTML mit CSS & XHTML von Kopf bis Fuß

OOA&D von Kopf bis Fuß

Head First Servlets and JSP

Head First EJB

Head First PMP (2007)

Head First Algebra (2007)

Head First Software Development (2007)

Der Inhalt (in der Übersicht)

Der Inhalt (jetzt ausführlich)

Einführung

Ihr Gehirn und OOA&D. Sie versuchen, etwas zu lernen, und Ihr Hirn tut sein Bestes, damit das Gelernte nicht hängen bleibt. Es denkt nämlich: »Wir sollten lieber ordentlich Platz für wichtigere Dinge lassen, z.B. für das Wissen, welche Tiere einem gefährlich werden könnten, oder dass es eine ganz schlechte Idee ist, nackt Snowboard zu fahren.« Tja, wie schaffen wir es nun, Ihr Gehirn davon zu überzeugen, dass Ihr Leben davon abhängt, etwas über objektorientierte Analyse und Design zu wissen?

Sauber entworfene Anwendungen rocken

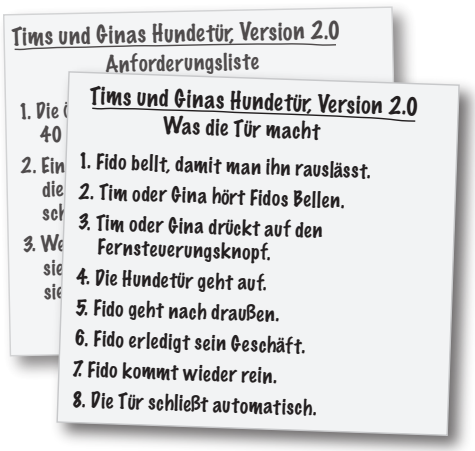
1 Hier fängt gute Software an

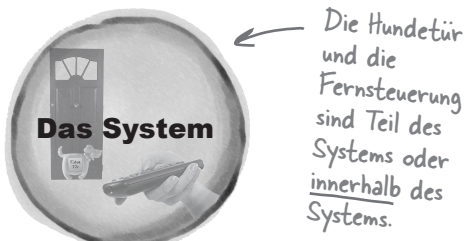
Wie schreibt man den nun *wirklich* gute Software? Es ist immer schwierig, den **Anfangspunkt zu finden**. Macht die Anwendung tatsächlich das, **was sie tun soll**? Und was ist mit Dingen wie doppeltem Code – das kann doch nicht gut sein, oder? In der Regel ist es ziemlich schwer, herauszufinden, **woran man als Erstes arbeiten soll**, und dabei gleichzeitig darauf zu achten, dass bei diesem Vorgang nichts anderes ruiniert wird. Aber Kopfzerbrechen ist unnötig. Wenn Sie mit diesem Kapitel durch sind, **werden Sie wissen, wie man gute Software schreibt**, und sind auf dem richtigen Weg, Ihre Art, Anwendungen zu entwickeln, dauerhaft zu verbessern. Und schließlich werden Sie verstehen, warum **OOA&D** ein Akronym ist, das man wirklich kennen sollte.

Anforderungen sammeln

2 Gebt Ihnen, was sie wollen

Jeder will zufriedene Kunden. Sie wissen bereits, der erste Schritt beim Schreiben guter Software ist sicherzustellen, dass sie macht, was der Kunde will. Aber wie findet man heraus, **was der Kunde wirklich will**? Und wie prüft man, ob der Kunde überhaupt *weiß*, was er wirklich will? Das ist der Punkt, an dem **gute Anforderungen** ins Spiel kommen, und in diesem Kapitel werden Sie lernen, wie Sie **Ihre Kunden zufriedenstellen**, indem Sie darauf achten, dass das, was Sie ausliefern, genau das ist, wonach der Kunde gefragt hat. Haben Sie die Arbeit abgeschlossen, sind all Ihre Projekte mit einer »Zufriedenheitsgarantie« ausgestattet. Und Sie sind auf dem besten Weg dahin, gute Software zu schreiben. Jedes Mal!

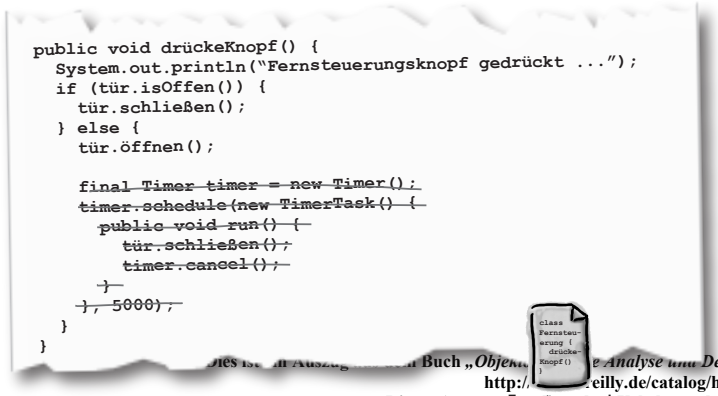




Anforderungen ändern sich

3 Ich liebe dich, du bist perfekt ... jetzt ändere dich endlich

Sie meinen, Sie haben genau das, was der Kunde wollte? Nicht so schnell ... Sie haben also mit Ihrem Kunden gesprochen, Anforderungen gesammelt, Ihre Anwendungsfälle geschrieben und eine Killer-Anwendung abgeliefert. Zeit für ein nettes kühles Bier, stimmt's? Stimmt ... bis sich Ihr Kunde überlegt, dass er eigentlich etwas **anderes will, als er Ihnen gesagt hat**. Was Sie gemacht haben, ist wunderbar, glauben Sie es mir, aber es ist **nicht mehr wirklich gut genug**. Im wahren Leben **ändern sich Anforderungen permanent**, und Sie müssen mit diesen Änderungen Schritt halten, um Ihre Kunden zufriedenzustellen.



Analyse

4 Ihre Software ins richtige Leben führen

Es ist Zeit, sich an richtige Anwendungen zu wagen.

Ihre Anwendung muss nicht nur auf Ihrer genau abgestimmten und perfekt eingerichteten Entwicklungsmaschine funktionieren. Ihre Anwendungen müssen funktionieren, wenn **echte Menschen sie verwenden**. In diesem Kapitel geht es darum, wie man sicherstellt, dass Software im **echten Leben** funktioniert. Sie werden lernen, wie eine **Textanalyse** den Anwendungsfall, an dem Sie gearbeitet haben, in Klassen und Methoden verwandelt, bei denen Sie sich darauf verlassen können, dass sie tun, was Ihre Kunden wollen. Und wenn Sie fertig sind, können auch Sie sagen: »Ich hab's geschaft! Meine Software ist **bereit fürs echte Leben**!«

Die richtige Welt

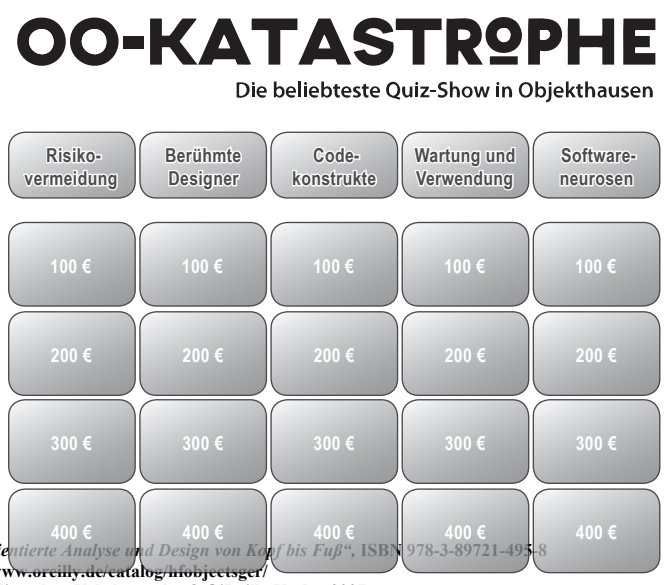
Gutes Design = flexible Software

5 (Teil 1)

Nichts bleibt jemals gleich

Veränderung ist unausweichlich. Egal ob Ihnen Ihre Software so gefällt, wie sie jetzt ist, morgen wird sie sich wahrscheinlich **ändern**. Und je schwerer Sie es machen, dass sich Ihre Software ändern kann, umso schwerer wird es, auf die **sich ändernden Anforderungen Ihres Kunden** zu reagieren. In diesem Kapitel werden wir einen alten Freund besuchen und versuchen, ein bestehendes Softwareprojekt zu ändern. Dabei werden wir uns ansehen, wie sich **kleine Änderungen in große Probleme verwandeln können**. Und wir werden tatsächlich ein Problem aufdecken, für dessen Lösung wir ein ZWEITEILIGES Kapitel brauchen!

5 (Intermezzo)



Gutes Design = flexible Software

5 (Teil 2) Geben Sie Ihrer Software ein 30-Minuten-Workout

Haben Sie sich jemals gewünscht, Sie wären flexibler?

Wenn Sie beim Ändern Ihrer Anwendung auf Probleme stoßen, bedeutet das wahrscheinlich, dass Ihre Software **flexibler und resistenter sein muss**. Um Ihre Anwendung etwas zu stärken, werden Sie etwas analysieren, eine Menge designen und lernen, wie OO-Prinzipien helfen können, **Ihre Anwendung aufzulockern**. Und im großen Finale werden Sie erfahren, wie **eine höhere Kohäsion Ihren Bindungen helfen kann**. Klingt interessant? Blättern Sie um, damit wir uns wieder der Reparatur dieser unflexiblen Anwendung widmen können.

Die richtig großen Probleme lösen

6 »Mein Name ist Art Vandelay ... Ich bin Architekt«

Es ist Zeit, etwas RICHTIG GROSSES aufzubauen. Sind Sie bereit?

Sie haben Massen von Werkzeugen in Ihrem OOA&D-Werkzeugkasten. Aber wie setzen Sie diese Werkzeuge ein, wenn Sie etwas **richtig Großes** aufbauen müssen? Na, vielleicht haben Sie es ja noch nicht erkannt, aber **Sie haben bereits alles, was Sie brauchen**, um große Probleme anzugehen. Wir werden ein paar neue Werkzeuge kennenlernen wie **Bereichsanalyse** und **Anwendungfallsdiagramme**, aber selbst diese neuen Werkzeuge basieren auf Dingen, die Sie bereits kennen – beispielsweise dass man auf den Kunden hören soll und verstehen muss, was man aufbaut, bevor man beginnt, Code zu schreiben. Machen Sie sich bereit ... es ist Zeit, Architekt zu spielen.

Architektur

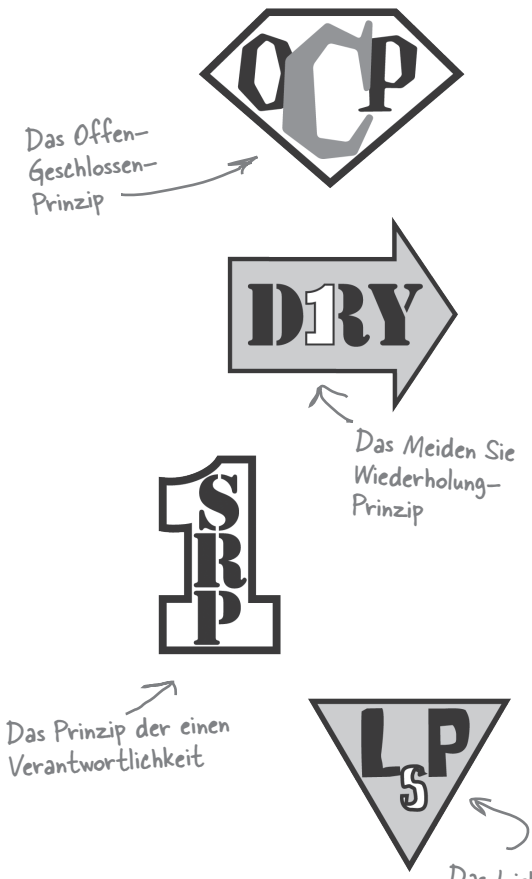
7 Ordnung ins Chaos bringen

Irgendwo müssen Sie anfangen, aber Sie wählen besser das *richtige* Irgendwo! Sie wissen, dass Sie Ihre Anwendung in viele kleine Probleme aufbrechen müssen. Aber das heißt bloß, dass Sie **VIELE** kleine Probleme haben. In diesem Kapitel werden wir Ihnen helfen, herauszufinden, **wo Sie anfangen sollten**, und sorgen dafür, dass Sie keine Zeit damit verschwenden, an den falschen Dingen zu arbeiten. Es ist Zeit, all diese **kleinen Teile** zu nehmen, die auf Ihrem Arbeitsplatz herumliegen, und herauszufinden, wie Sie sie in eine **wohlgeordnete, sauber entworfene Anwendung** verwandeln. Unterwegs werden Sie noch etwas zu den allmächtigen **drei Fs der Architektur** lernen und erfahren, warum **Risiko** mehr als ein cooles Kriegsspiel aus den 80er-Jahren ist.

Design-Prinzipien

8 Originalität wird überschätzt

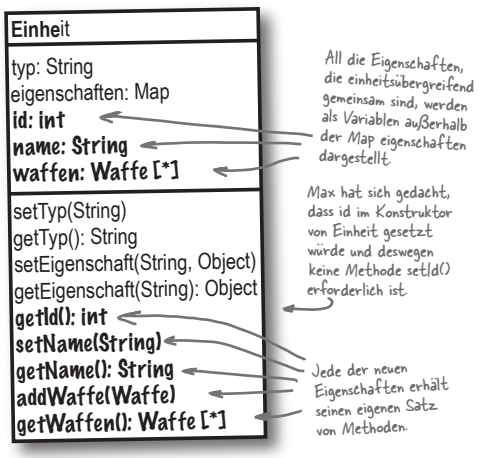
Nachahmung ist die aufrichtigste Form, keine Dummheiten zu machen. Es gibt nichts, was so befriedigend ist wie die Entdeckung einer vollständig neuen und originellen Lösung für ein Problem, das einen seit Tagen quält – bis man entdeckt, dass ein anderer schon **das gleiche Problem gelöst hat**, lange vor einem, und seine Sache dabei noch besser gemacht hat als man selbst! In diesem Kapitel werden wir uns einige **Design-Prinzipien** anschauen, die andere über die Jahre gefunden haben, und wie sie Sie zu einem besseren Programmierer machen können. Legen Sie Ihren Anspruch ab, »es auf meine Weise zu machen«. In diesem Kapitel geht es darum, **den klügeren, schnelleren Weg zu nehmen.**



Iteration und Tests

9 Die Software ist immer noch für den Kunden

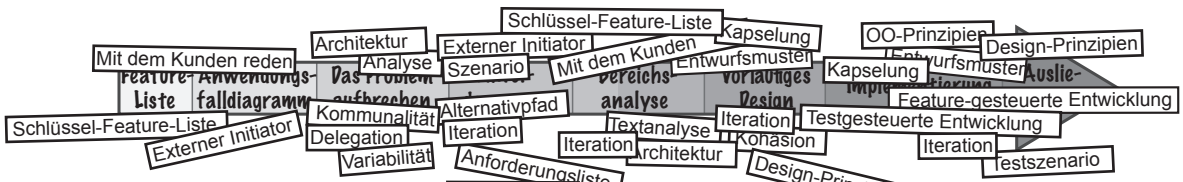
Es ist Zeit, dem Kunden zu zeigen, wie viele Gedanken Sie sich wirklich machen. Nörgelnde Chefs? Besorgte Kunden? Teilhaber, die ständig fragen: »Wird es rechtzeitig fertig?« Der ganze sauber entworfene Code wird Ihre Kunden nicht zufriedenstellen. Sie müssen **Ihnen etwas zeigen, das funktionieriert**. Und da Sie jetzt einen stabilen Werkzeugkasten für die OO-Programmierung haben, ist es Zeit, dass Sie lernen, wie Sie **dem Kunden beweisen können**, dass Ihre Software funktioniert. In diesem Kapitel werden wir zwei Wege kennenlernen, **tiefer** in die Funktionalität Ihrer Software **einzutauchen** und dem Kunden das angenehme Gefühl zu vermitteln, das ihn sagen lässt: ***Ja. Du hast definitiv den richtigen Entwickler für diesen Job!***



Der OOA&D-Lebenszyklus

10 Alles zusammenfügen

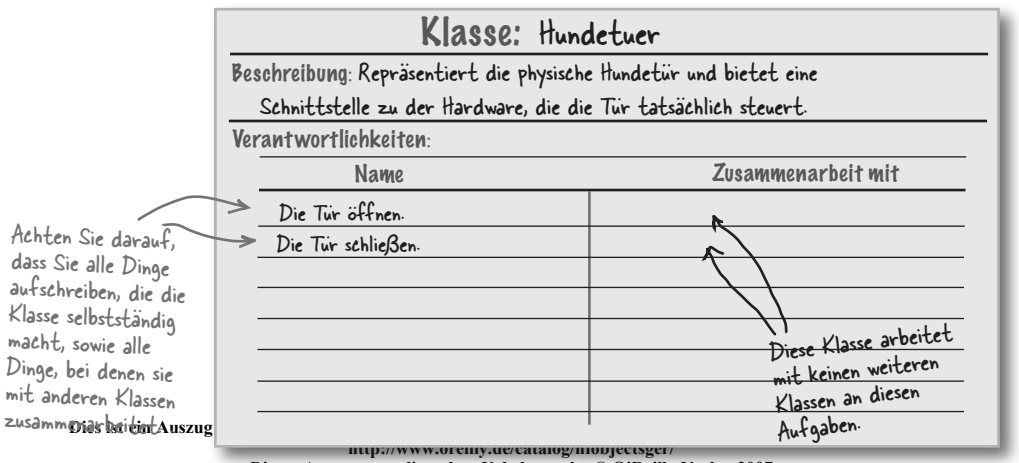
Sind wir schon angekommen? Wir haben eine Menge an den einzelnen Möglichkeiten gearbeitet, Ihre Software zu verbessern, aber jetzt ist es Zeit, **alles zusammenzufügen**. Das ist doch das, worauf Sie gewartet haben: Wir werden **alles** nehmen, was Sie gelernt haben, und Ihnen zeigen, wie das alles Teil eines **einzigen Prozesses** ist, den Sie immer wieder einsetzen können, um **gute Software zu schreiben**.



Anhang A: Was übrig bleibt

Die Top Ten der Themen, die wir nicht behandelt haben

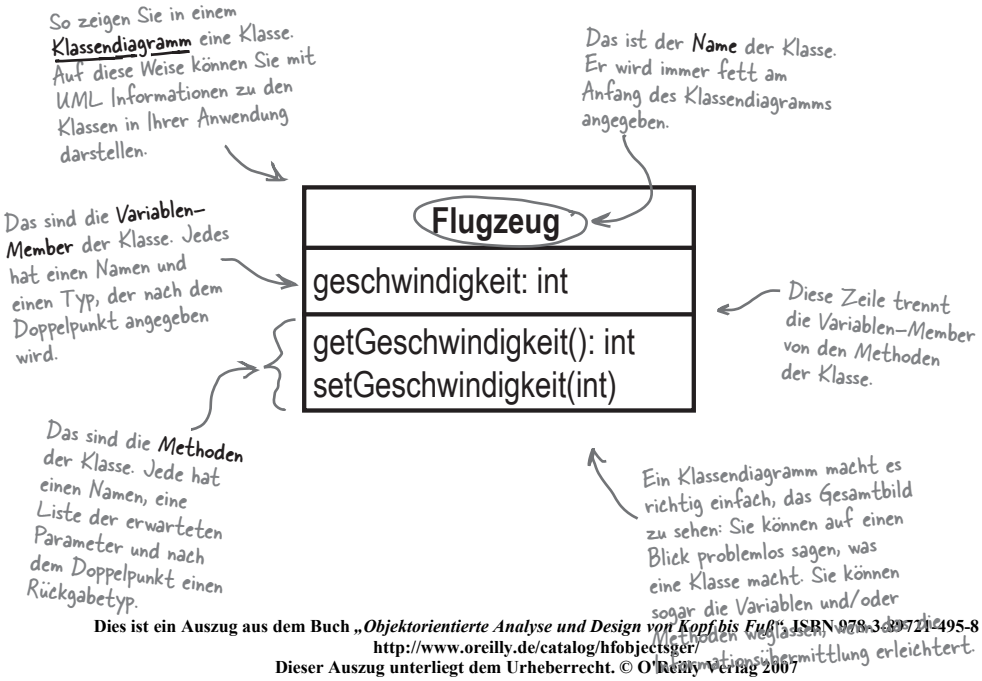
Glauben Sie es oder nicht: Es kommt noch mehr. Ja. Auch mit über 500 Seiten im Rücken gibt es immer noch Dinge, die wir nicht reinstopfen konnten. Selbst wenn diese abschließenden Top-Ten-Themen nicht mehr als eine Erwähnung verdienen, wollten wir Sie nicht ohne ein paar zusätzliche Informationen über jedes von ihnen aus Objekthausen entlassen. Kopf hoch. Schließlich haben Sie dann etwas mehr, über das Sie während der Werbeunterbrechungen bei OO-KATASTROPHE! reden können ... und wer steht nicht gelegentlich auf eine nette OOA&D-Plauderei?



Anhang B: Anleitung für Objekthausen

B Die OO-Sprache sprechen

Sind Sie bereit für für eine Reise in ein fremdes Land? Objekthausen ist eine seltsame Stadt, deren Umgangs- und Verkehrsregeln Ihnen beim ersten Aufenthalt vielleicht merkwürdig vorkommen könnten. Sie fühlen sich ausgeschlossen, weil das **Erben** scheinbar nur bei den Anderen funktioniert? Die häufige Erwähnung von **Polymorphie** lässt bei Ihnen ernsthafte Zweifel an den Moralvorstellung der Einheimischen aufsteigen? Dann sollten Sie zur **Akklimatisierung** zunächst vielleicht diesen Anhang lesen, um sich mit den Sitten und der Sprache von Objekthausen **vertraut zu machen**. Sie müssen sich keine Sorgen machen. Es wird nicht lange dauern, und bevor Sie es merken, werden Sie die **OO-Sprache sprechen**, als würden Sie schon seit Jahren in den sauber entworfenen Gebieten von Objekthausen leben.



3 Anforderungen ändern sich

Ich liebe dich, du bist perfekt ... jetzt ändere dich endlich

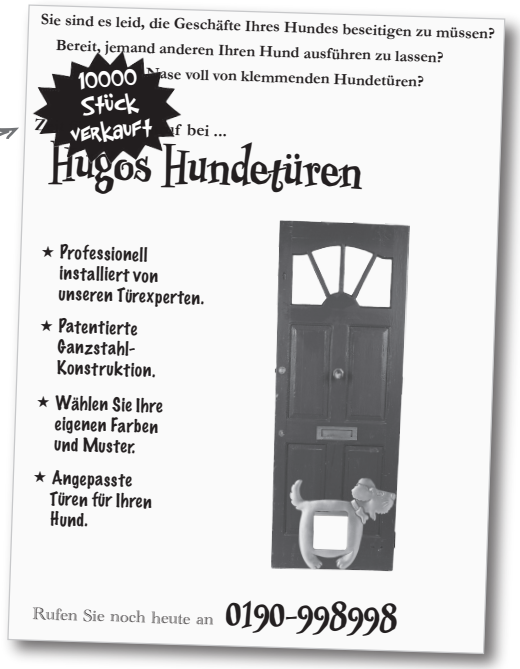
Sie meinen, Sie haben genau das, was der Kunde wollte?

Nicht so schnell ... Sie haben also mit Ihrem Kunden gesprochen, Anforderungen gesammelt, Ihre Anwendungsfälle geschrieben und eine Killer-Anwendung abgeliefert. Zeit für ein nettes kühles Bier, stimmt's? Stimmt ... bis sich Ihr Kunde überlegt, dass er eigentlich etwas **anderes will, als er Ihnen gesagt hat**. Was Sie gemacht haben, ist wunderbar, glauben Sie es mir, aber es ist **nicht mehr wirklich gut genug**. Im wahren Leben **ändern sich Anforderungen permanent**, und Sie müssen mit diesen Änderungen Schritt halten, um Ihren Kunden zufriedenzustellen.

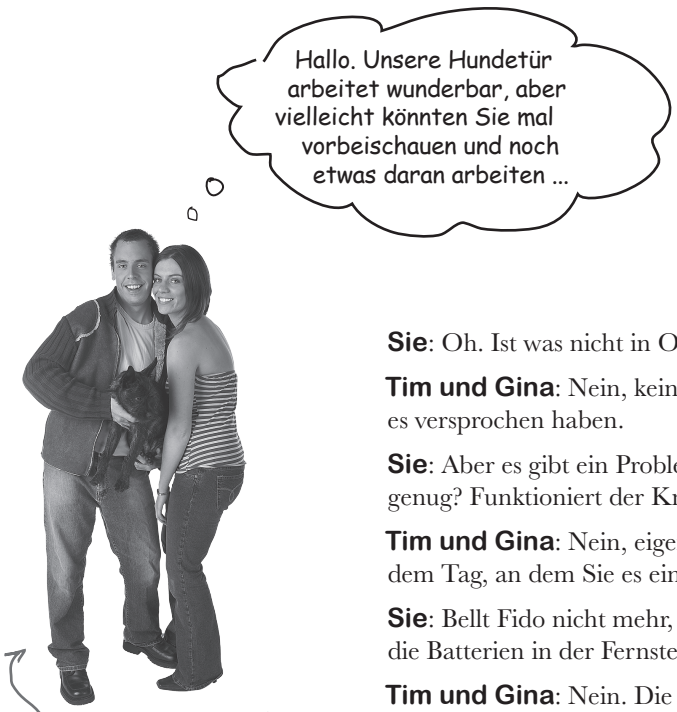
Sie sind ein Held!

Sie nuckeln an einer leckeren Piña Colada, die Sonne brennt auf Sie herab, und aus Ihrer Badehose ragt eine Rolle 100-€-Scheine ... das ist das Leben des Programmierers, der gerade Hugos Hundetüren zu einem erfolgreichen Unternehmen gemacht hat. Die Tür, die Sie für Tim und Gina gebaut haben, war ein gewaltiger Erfolg, und Hugo verkauft sie jetzt an Kunden auf der ganzen Welt.

Hugo macht eine ganze Menge Kohle mit Ihrem Code.



Aber dann klingelt das Telefon ...



Tim und Gina unterbrechen fröhlich Ihren wohlverdienten Urlaub.

Sie: Oh. Ist was nicht in Ordnung?

Tim und Gina: Nein, keineswegs. Die Tür funktioniert genau so, wie Sie es versprochen haben.

Sie: Aber es gibt ein Problem, stimmt's? Schließt die Tür nicht schnell genug? Funktioniert der Knopf auf der Fernsteuerung nicht?

Tim und Gina: Nein, eigentlich ... es funktioniert alles genauso gut wie an dem Tag, an dem Sie es eingebaut und uns vorgeführt haben.

Sie: Bellt Fido nicht mehr, wenn er nach draußen will? Mmm, haben Sie die Batterien in der Fernsteuerung geprüft?

Tim und Gina: Nein. Die Tür ist absolut wunderbar. Wir haben uns einfach nur ein paar Änderungen überlegt, die Sie für uns einbauen könnten ...

Sie: Aber wenn alles funktioniert, was ist dann das Problem?

Wir sind es beide leid, immer darauf zu achten, was Fido gerade macht. Manchmal hören wir ihn einfach nicht bellen, und er pinkelt drinnen.

Und ständig verlegen wir diese Fernsteuerung oder lassen sie in einem anderen Zimmer. Außerdem habe ich keine Lust mehr, ständig einen Knopf drücken zu müssen, um die Tür zu öffnen.

Tims und Ginas Hundetür, Version 2.0

Was die Tür (aktuell) macht

1. Fido bellt, damit man ihn rauslässt.
2. Tim oder Gina hört Fidos Bellen.
3. Tim oder Gina drückt auf den Fernsteuerungsknopf.
4. Die Hundetür geht auf.
5. Fido geht nach draußen.
6. Fido erledigt sein Geschäft.
   - 6.1. Die Tür schließt automatisch.
   - 6.2. Fido bellt, damit er wieder reingelassen wird.
   - 6.3. Tim oder Gina hört Fidos erneutes Bellen.
   - 6.4. Tim oder Gina drückt auf den Fernsteuerungsknopf.
   - 6.5. Die Hundetür geht wieder auf.
7. Fido kommt (wieder) rein.
8. Die Tür schließt automatisch.

Was wäre, wenn sich die Hundetür **automatisch** öffnete, wenn Fido sie anbellt? Dann müssten wir gar nichts mehr tun, um ihn rauszulassen! Wir haben das durchgesprochen und halten es für eine **TOLLE** Idee!

Zurück ans Reißbrett

Machen wir uns an die Arbeit, um Tims und Ginas Hundetür wieder mal zu verbessern. Wir müssen eine Möglichkeit finden, die Tür zu öffnen, wenn Fido bellt. Beginnen wir damit ...

Einen Augenblick ... das ist absolut nervend! Wir haben ihnen bereits eine **funktionierende** Tür gebaut, und sie haben gesagt, sie wäre **in Ordnung**. Und jetzt sollen wir weitere Änderungen an der Tür vornehmen, nur weil sie eine neue Idee haben?

Der Kunde hat immer recht

Auch wenn sich die Anforderungen ändern, müssen Sie bereit sein, Ihre Anwendung zu aktualisieren und sicherzustellen, dass sie funktioniert, wie es der Kunde erwartet. Wenn der Kunde neue Bedürfnisse hat, ist es Ihre Aufgabe, die Anwendung so zu ändern, dass sie diese neuen Bedürfnisse befriedigt.

Hugo gefällt es, wenn das eintritt, weil er Tim und Gina die Änderungen in Rechnung stellen kann, die Sie machen.

KOPF-NUSS

Sie haben gerade eine Konstante bei der Analyse und dem Design von Software entdeckt. Was, glauben Sie, ist diese Konstante?

Die Konstante bei Softwareanalyse und -design*

Okay, worauf können Sie sich immer verlassen, wenn Sie Software schreiben?

Ganz egal wo Sie arbeiten, woran Sie arbeiten oder in welcher Sprache Sie programmieren, was ist die eine Konstante, die Sie immer begleiten wird?

VERÄNDERUNG

(verwenden Sie einen Spiegel, um die Antwort zu sehen)

Egal wie gut Sie eine Anwendung entwerfen, mit der Zeit wird diese Anwendung stetig wachsen und sich verändern. Sie werden neue Lösungen für Probleme entdecken, Programmiersprachen entwickeln sich, oder nette Kunden werfen verrückte neue Anforderungen auf, die Sie zwingen, funktionierende Anwendungen zu »reparieren«.

Spitzen Sie Ihren Bleistift

Anforderungen ändern sich permanent ... manchmal mitten in einem Projekt und manchmal, wenn Sie meinen, alles sei fertig. Schreiben Sie einige Gründe auf, aus denen sich die Anforderungen bei den Anwendungen ändern könnten, an denen Sie aktuell arbeiten.

Mein Kunde hat entschieden, dass die Anwendung anders arbeiten soll.

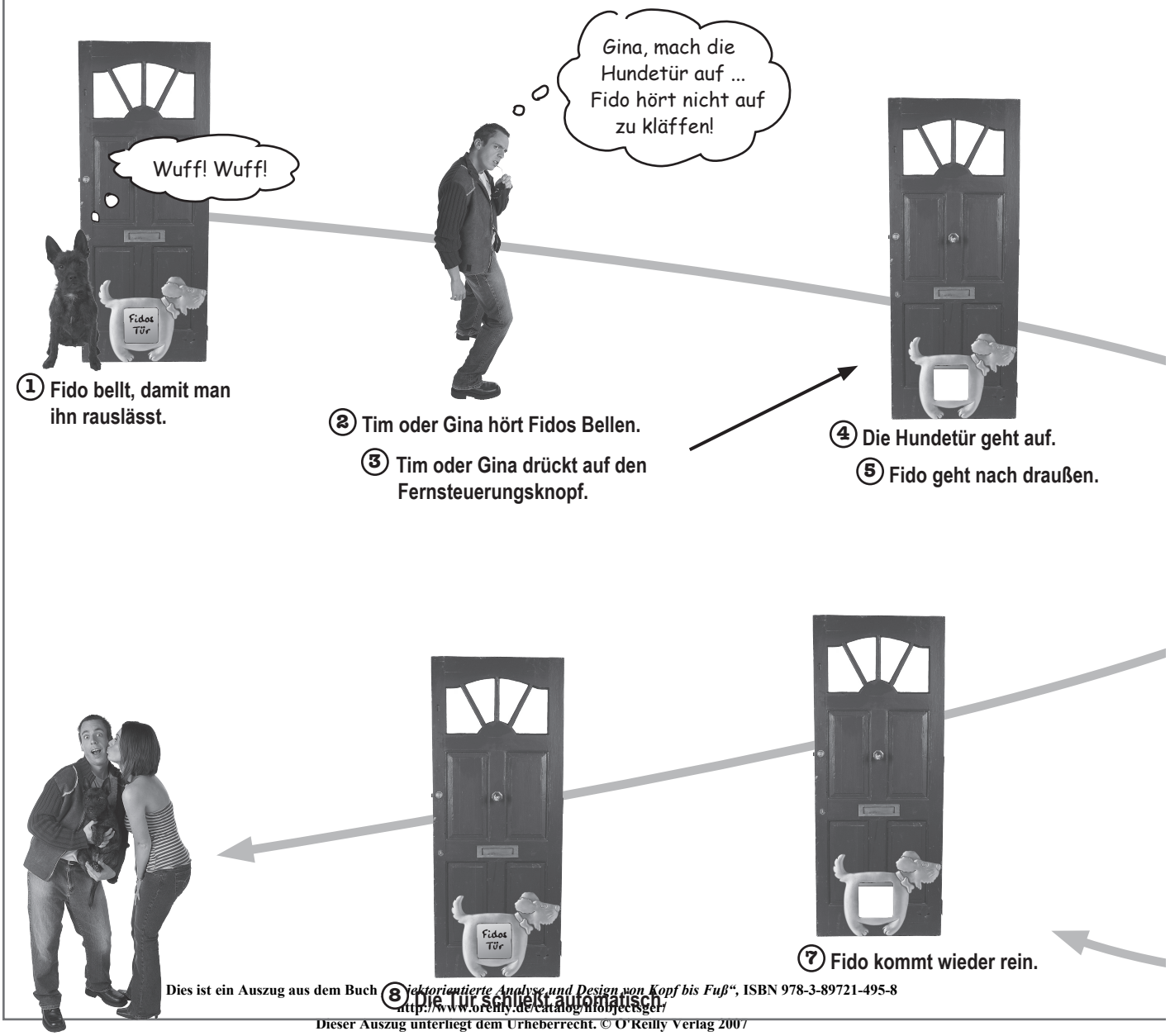
Mein Boss meint, die Anwendung wäre besser eine Web– als eine Desktop–Anwendung.

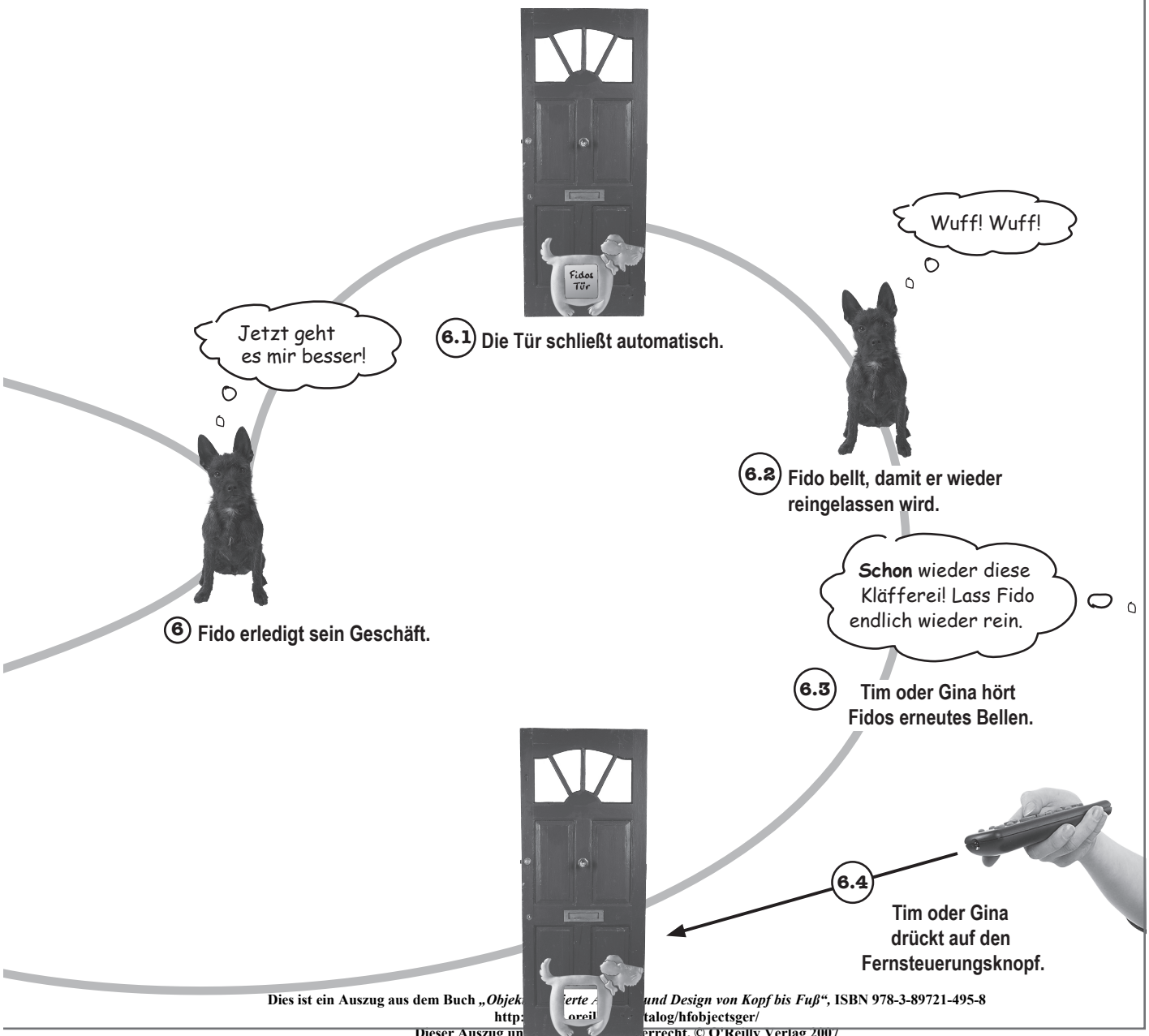
Anforderungen ändern sich immer. Aber wenn Sie gute Anwendungsfälle haben, können Sie Ihre Software in der Regel schnell ändern, um sie an diese neuen Anforderungen anzupassen.

*Wenn Sie Entwurfsmuster von Kopf bis Fuß gelesen haben, könnte Ihnen diese Seite bekannt vorkommen. Dude, wir haben bei Entwurfsmustern gute Arbeit geleistet, dass wir uns entschlossen haben, einfach die Ideen zu klauen und hier und da ein paar Dinge zu VERÄNDERN. Danke, Beth und Eric!

Fügen Sie Tims und Ginas Hundetür ein Bellerkennung hinzu.

Aktualisieren Sie das Diagramm, um einen Alternativpfad hinzuzufügen, bei dem Fido bellt, Hugos neue Bellerkennung Fido hört und automatisch die Tür öffnet. Die Fernsteuerung sollte aber weiterhin funktionieren. Entfernen Sie also nichts aus dem Diagramm. Fügen Sie einfach einen weiteren Pfad hinzu, bei dem Fidos Bellen die Tür öffnet.

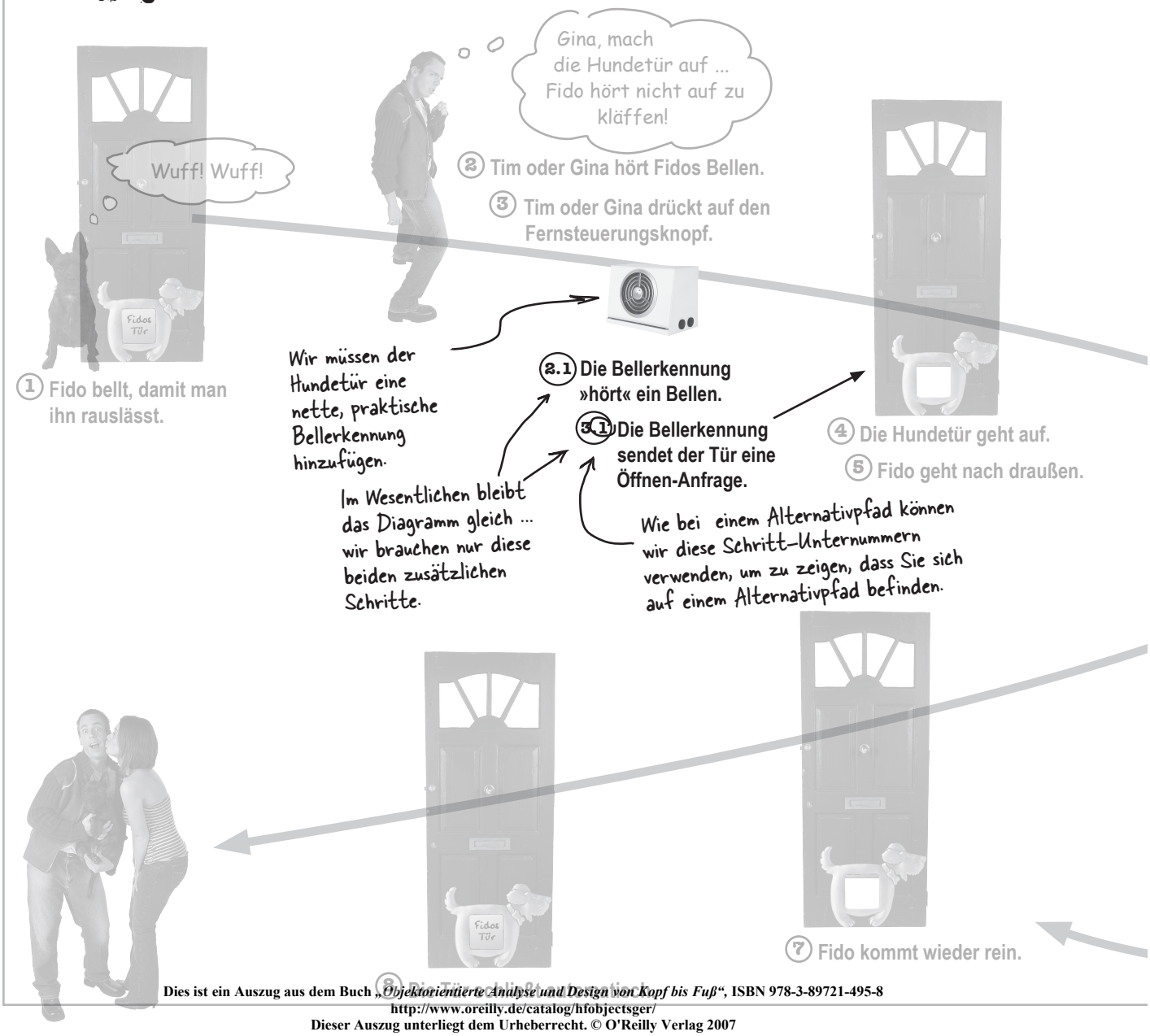


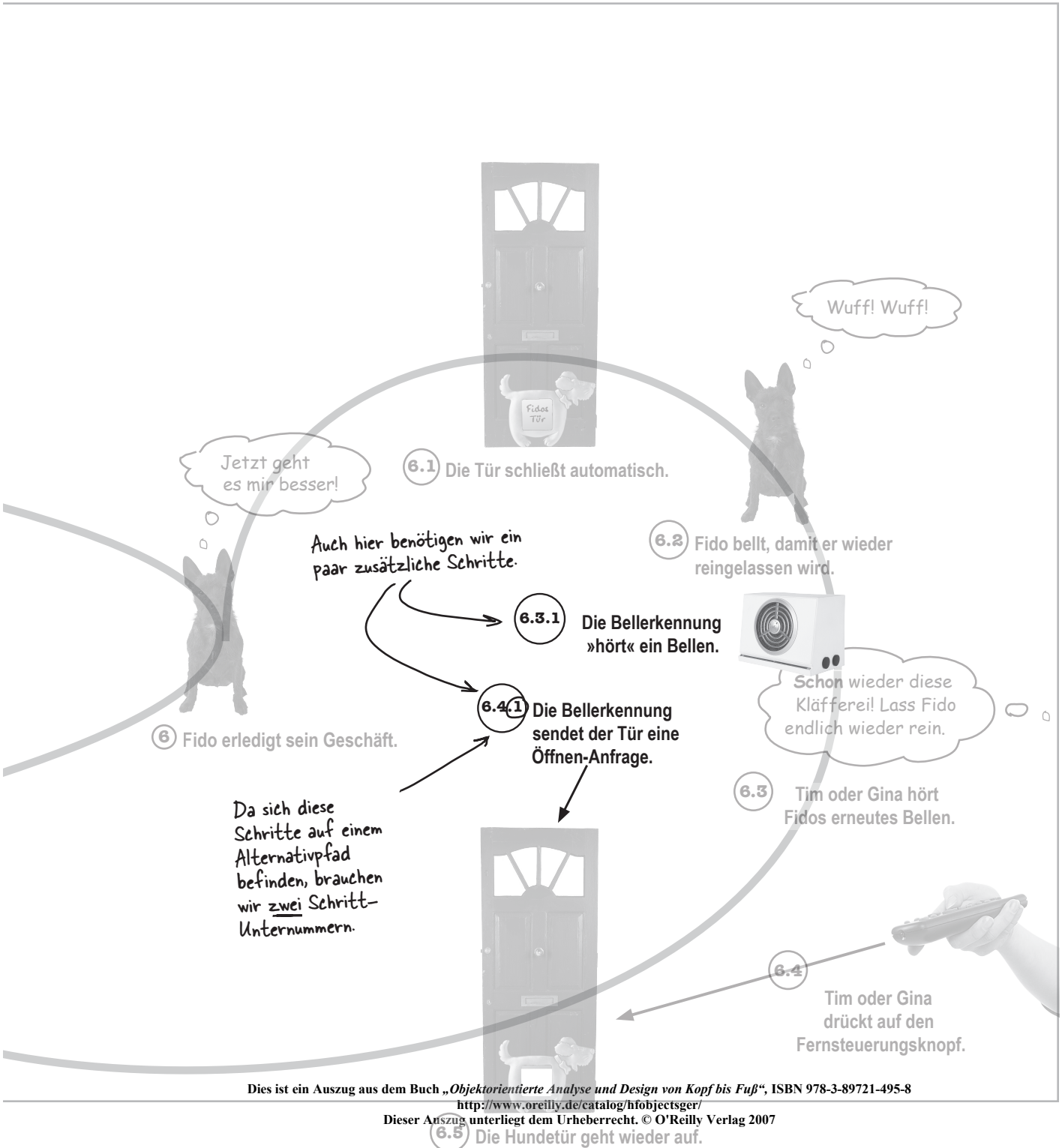


Lösungen zu den Übungen

Hugo hat Hardware erfunden, die ein Bellen erkennt, aber Sie müssen herausfinden, wie Sie diese neue Hardware im Hundetür-System verwenden können.

So haben wir Tims und Ginas Problem gelöst und ihre Hundetür mit Bellerkennung implementiert. Schauen Sie, ob Sie dem Diagramm die gleichen Dinge hinzugefügt haben.





Aber jetzt ist mein Anwendungsfall vollkommen verwirrend. All diese Alternativpfade erschweren es festzustellen, was da eigentlich vor sich geht.

Optionale Pfade? Alternative Pfade? Wer weiß das schon?

Tims und Ginas Hundetür, Version 2.1
Was die Tür macht

1. Fido bellt, damit man ihn rauslässt.
2. Tim oder Gina hört Fidos Bellen.
   - 2.1. Die Bellerkennung »hört« ein Bellen.
3. Tim oder Gina drückt auf den Fernsteuerungsknopf.
   - 3.1. Die Bellerkennung sendet der Tür eine Öffnen-Anfrage.
4. Die Hundetür geht auf.
5. Fido geht nach draußen.
6. Fido erledigt sein Geschäft.
   - 6.1. Die Tür schließt automatisch.
   - 6.2. Fido bellt, damit er wieder reingelassen wird.
   - 6.3. Tim oder Gina hört Fidos erneutes Bellen.
     - 6.3.1. Die Bellerkennung »hört« wieder ein Bellen.
   - 6.4. Tim oder Gina drückt auf den Fernsteuerungsknopf.
     - 6.4.1. Die Bellerkennung sendet der Tür eine Öffnen-Anfrage.
   - 6.5. Die Hundetür geht wieder auf.
7. Fido kommt wieder rein.
8. Die Tür schließt automatisch.

Jetzt gibt es zwei Alternativschritte für Nummer 2 und 3.

Die Punkte werden als Unterschritte aufgeführt, bieten aber eigentlich einen vollständig anderen Pfad durch den Anwendungsfall.

Diese Unterschritte stellen eine zusätzliche Menge von Schritten, die befolgt werden können ...

... aber diese Schritte sind eigentlich ein anderer Weg durch den Anwendungsfall.

Selbst Alternativschritte haben jetzt Alternativschritte.

", ISBN 978-3-89721-495-8

Ich denke, dieser Anwendungsfall ist verwirrend. Das sieht so aus, als würden Tim und Gina Fido **immer** bellen hören, die Bellerkennung aber nur **manchmal**. Aber das ist nicht das, was Tim und Gina wollen ...

Sehen Sie, was Gerald meint? Tims und Ginas Idee war, dass sie nicht mehr auf Fidos Bellen hören müssen.

Tims und Ginas Hundetür, Version 2.1
Was die Tür macht

1. Fido bellt, damit man ihn rauslässt.
2. Tim oder Gina hört Fidos Bellen.
   - 2.1. Die Bellerkennung »hört« ein Bellen.
3. Tim oder Gina drückt auf den Fernsteuerungsknopf.
   - 3.1. Die Bellerkennung sendet der Tür eine Öffnen-Anfrage.
4. Die Hundetür geht auf.
5. Fido geht nach draußen.
6. Fido erledigt sein Geschäft.
   - 6.1. Die Tür schließt automatisch.
   - 6.2. Fido bellt, damit er wieder reingelassen wird.
   - 6.3. Tim oder Gina hört Fidos erneutes Bellen.
     - 6.3.1. Die Bellerkennung »hört« wieder ein Bellen.
   - 6.4. Tim oder Gina drückt auf den Fernsteuerungsknopf.
     - 6.4.1. Die Bellerkennung sendet der Tür eine Öffnen-Anfrage.
   - 6.5. Die Hundetür geht wieder auf.
7. Fido kommt wieder rein.
8. Die Tür schließt automatisch.

Im neuen Anwendungsfall wollen wir eigentlich sagen, dass entweder Schritt 2 oder Schritt 2.1 eintritt ....

... und dann entweder Schritt 3 oder Schritt 3.1 folgt.

Hier erfolgt entweder Schritt 6.3 oder 6.3.1 ...

... und dann erfolgt entweder 6.4 oder 6.4.1.

Sie müssen den Anwendungsfall verstehen

Wenn ein Anwendungsfall für Sie verwirrend ist , *können Sie ihn einfach umschreiben*. Es gibt jede Menge unterschiedlicher Arten, auf die Anwendungsfälle geschrieben werden. Aber das Wichtige dabei ist, dass Sie, Ihr Team und die Leute den Anwendungsfall verstehen, denen Sie ihn erklären müssen. Schreiben wir also den Anwendungsfall von Seite 121 um, damit er nicht mehr so verwirrend ist.

Wir haben die Schritte, die an Stelle der Schritte des Hauptpfads erfolgen können, nach rechts verschoben.

Tims und Ginas Hundetür, Version 2.2
Was die Tür macht

Jetzt haben wir eine Beschriftung hinzugefügt, die uns sagt, dass die Schritte auf der linken Seite der Hauptpfad sind.

Hauptpfad

1. Fido bellt, damit man ihn rauslässt.
2. Tim oder Gina hört Fidos Bellen.
3. Tim oder Gina drückt auf den Fernsteuerungsknopf.
4. Die Hundetür geht auf.
5. Fido geht nach draußen.
6. Fido erledigt sein Geschäft.
   - 6.1. Die Tür schließt automatisch.
   - 6.2. Fido bellt, damit er wieder reingelassen wird.
   - 6.3. Tim oder Gina hört Fidos erneutes Bellen.
   - 6.4. Tim oder Gina drückt auf den Fernsteuerungsknopf.
   - 6.5. Die Hundetür geht wieder auf.
7. Fido kommt wieder rein.
8. Die Tür schließt automatisch.

Alternativpfade

- 2.1. Die Bellerkennung »hört« ein Bellen.
- 3.1. Die Bellerkennung sendet der Tür eine Öffnen-Anfrage.
- 6.3.1. Die Bellerkennung »hört« wieder ein Bellen.
- 6.4.1. Die Bellerkennung sendet der Tür eine Öffnen-Anfrage.

Wenn es nur einen Schritt gibt, nutzen wir immer diesen Schritt, wenn wir den Anwendungsfall durchlaufen.

Jetzt ist das etwas klarer: Wir können Schritt 2 ODER Schritt 2.1 und dann Schritt 3 ODER Schritt 3.1 verwenden.

Diese Unterschritte sind optional ... Sie können sie nutzen, müssen es aber nicht. Aber sie stehen immer noch auf der linken Seite, weil sie keine Schritte im Hauptpfad ersetzen.

Diese Schritte auf der rechten Seite können die Schritte 6.3 und 6.4 ersetzen. Sie können nur einen der Schritte wählen, wenn Sie den Anwendungsfall durchlaufen: entweder den Schritt auf der linken ODER den Schritt auf der rechten Seite.

Egal wie Sie diesen Anwendungsfall durchlaufen, Sie enden immer bei Schritt 8 des Hauptpfads.

Hervorragende Idee!

Der Hauptpfad sollte das sein, was in der Regel passieren soll. Da Tim und Gina wahrscheinlich möchten, dass sich die Bellerkennung öfter um Fido kümmert als sie selbst mit ihrer Fernbedienung, sollten wir diese Schritte in unseren Hauptpfad stecken:

Tims und Ginas Hundetür, Version 2.3
Was die Tür macht

Hauptpfad

1. Fido bellt, damit man ihn rauslässt.
2. Die Bellerkennung »hört« ein Bellen.
3. Die Bellerkennung sendet der Tür eine Öffnen-Anfrage.
4. Die Hundetür geht auf.
5. Fido geht nach draußen.
6. Fido erledigt sein Geschäft.
   - 6.1. Die Tür schließt automatisch.
   - 6.2. Fido bellt, damit er wieder reingelassen wird.
   - 6.3. Die Bellerkennung »hört« wieder ein Bellen.
   - 6.4. Die Bellerkennung sendet der Tür eine Öffnen-Anfrage.
   - 6.5. Die Hundetür geht wieder auf.
7. Fido kommt wieder rein.
8. Die Tür schließt automatisch.

Die Schritte, die die Bellerkennung betreffen, befinden sich jetzt auf dem Hauptpfad statt auf einem Alternativpfad.

Alternativpfad

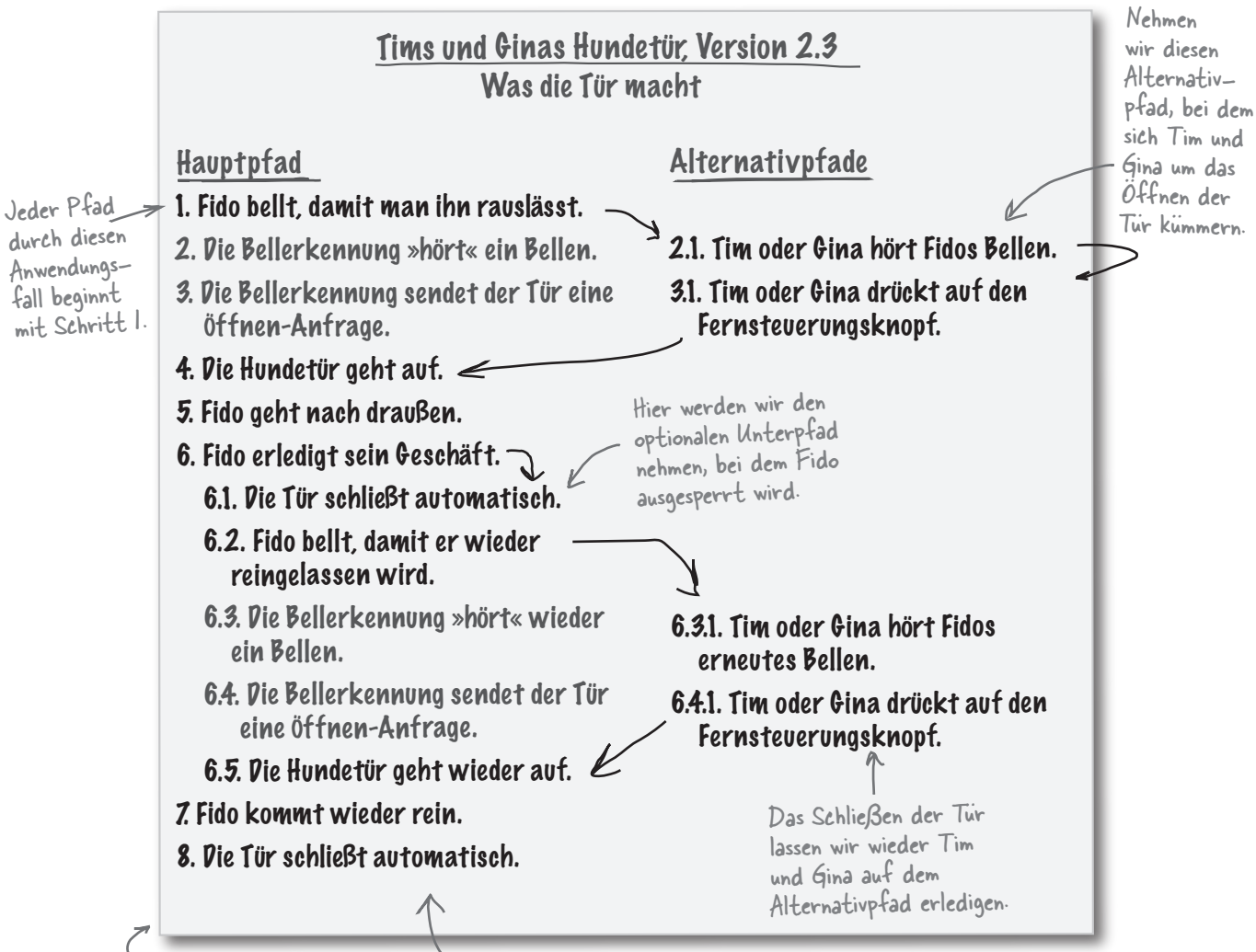
- 2.1. Tim oder Gina hört Fidos Bellen.
- 3.1. Tim oder Gina drückt auf den Fernsteuerungsknopf.

Normalerweise möchten Tim und Gina die Fernbedienung nicht verwenden. Die Schritte, die die Fernbedienung betreffen, eignen sich also besser für einen Alternativpfad.

- 6.3.1. Tim oder Gina hört Fidos erneutes Bellen.
- 6.4.1. Tim oder Gina drückt auf den Fernsteuerungsknopf.

Vom Start zum Ziel: ein bestimmtes Szenario

Mit all den Alternativpfaden in unserem neuen Anwendungsfall gibt es viele verschiedene Wege, Fido nach draußen aufs Örtchen und dann wieder hinein zu befördern. Dies ist ein bestimmter Weg durch den Anwendungsfall:



Es gibt keine Dummen Fragen

F: Ich verstehe, was der Hauptpfad eines Anwendungsfalls ist, aber können Sie noch mal erklären, was ein Alternativpfad ist?

A: Ein Alternativpfad sind ein oder mehrere Schritte eines Anwendungsfalls, die optional sind oder alternative Wege bieten, den Anwendungsfall zu durchlaufen. Alternativpfade können *zusätzliche* Schritte sein, die dem Hauptpfad hinzugefügt werden, oder Schritte anbieten, die es Ihnen ermöglichen, das Ziel auf *ganz andere Weise* zu erreichen als über Teile des Hauptpfads.

F: Wenn Fido nach draußen geht und ausgesperrt wird, ist das also Teil eines Alternativpfads, stimmt's?

A: Richtig. Im Anwendungsfall sind die Schritte 6.1, 6.2, 6.3, 6.4 und 6.5 ein Alternativpfad. Diese Schritte sind *zusätzliche* Schritte, die das System eventuell durchlaufen muss und die nur benötigt werden, wenn Fido ausgesperrt wird. Aber es ist ein Alternativpfad, weil Fido nicht *immer* ausgesperrt wird – das System könnte von Schritt 6 auch direkt zu Schritt 7 gehen.

F: Und dafür nutzen wir Unterschritte wie 6.1 und 6.2?

A: Genau. Weil ein Alternativpfad, der zusätzliche Schritte hat, nur ein Satz von Schritten ist, der als *Teil eines* anderen Schritts des Hauptpfads des Anwendungsfalls eintreten kann. Wenn Fido ausgesperrt wird, sind die Schritte des Hauptpfads 6 und 7. Der Alternativpfad beginnt also bei Schritt 6.1 und geht bis Schritt 6.5. Diese sind ein optionaler Teil von Schritt 6.

F: Und wie nennen Sie das, wenn es zwei *verschiedene* Wege durch einen Teil eines Anwendungsfalls gibt?

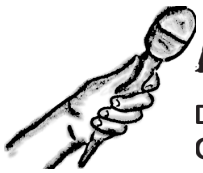
A: Hm. Eigentlich ist das nur eine andere Art von Alternativpfad. Wenn Fido bellt, gibt es einen Pfad, bei dem Tim und Gina Fidos Bellen hören und die Tür für ihn öffnen, und einen Pfad, bei dem die Bellerkennung ein Bellen hört und die Tür öffnet. Aber das System ist so entworfen, dass nur einer von beiden, nicht beide eintreten – entweder öffnet die Fernsteuerung die Tür oder die Bellerkennung.

F: Kann es in einem Anwendungsfall mehrere Alternativpfade geben?

A: Natürlich. Sie können Alternativpfade haben, die zusätzliche Schritte bieten, und mehrere Wege, um von der Anfangsbedingung zur Endbedingunge zu kommen. Sie können sogar einen Alternativpfad haben, der den Anwendungsfall vorzeitig beendet ... aber so etwas Kompliziertes brauchen wir für die Hundetür von Tim und Gina nicht.

Ein vollständiger Pfad durch einen Anwendungsfall vom ersten bis zum letzten Schritt wird als ein Szenario bezeichnet.

Die meisten Anwendungsfälle haben mehrere unterschiedliche Szenarien, aber diese teilen sich immer das gleiche Benutzerziel.



Anwendungsfälle im Gespräch

Diese Woche:
Geständnisse eines Alternativpfads

Von Kopf bis Fuß: Hallo, Alternativpfad. Wir haben gehört, dass Sie in letzter Zeit ziemlich unglücklich sind. Erzählen Sie uns, was los ist.

Alternativpfad: Ich fühle mich manchmal einfach ausgeschlossen. Sehen Sie, ohne mich kann man kaum einen ordentlichen Anwendungsfall zusammenstellen, und trotzdem scheint es, als würde ich die ganze Zeit ignoriert.

Von Kopf bis Fuß: Ignoriert? Aber Sie haben doch gerade gesagt, dass Sie Teil fast jedes Anwendungsfalls sind. Für mich hört sich das so an, als wären Sie eigentlich ziemlich wichtig.

Alternativpfad: Sicher. *Vielleicht* klingt das so. Aber selbst wenn ich Teil eines Anwendungsfalls bin, kann ich für einige andere Schritte übersprungen werden. Das stinkt mir ... das ist so, als wäre ich überhaupt nicht da!

Von Kopf bis Fuß: Können Sie uns ein Beispiel geben?

Alternativpfad: Gerade neulich war ich Teil eines Anwendungsfalls für den Erwerb einer CD bei diesem tollen neuen Onlinestore Musikologie. Ich war so aufgeregt ... aber dann hat sich herausgestellt, dass ich mich lediglich um die Situation kümmern sollte, die eintritt, wenn die Kreditkarte des Kundens abgelehnt wird.

Von Kopf bis Fuß: Na, das klingt doch nach einer richtig wichtigen Aufgabe! Was ist da das Problem?

Alternativpfad: Hm, ich nehme mal an, dass das wichtig ist. Aber ich werde immer übersprungen. Es scheint, als könnte jeder problemlos CDs bestellen, weil alle Kreditkarten akzeptiert werden. Ich war zwar Teil des Anwendungsfalls, aber nicht Teil der häufigsten Szenarien.

Von Kopf bis Fuß: Ah. Ich verstehe. Sie waren also nie beteiligt, es sei denn, es wurde eine Kreditkarte abgelehnt.

Alternativpfad: Genau! Die Typen aus der Buchhaltung und der Security liebten mich und haben die ganze Zeit davon geschwärmt, wie wichtig ich für das Unternehmen sei. Aber wer will schon die ganze Zeit untätig rumsitzen?

Von Kopf bis Fuß: So langsam verstehe ich. Aber trotzdem helfen Sie doch dem Anwendungsfall, oder? Selbst wenn Sie nicht die ganze Zeit verwendet werden, passiert es doch gelegentlich, dass Sie aufgerufen werden.

Alternativpfad: Das ist wahr. Wir haben alle das gleiche Ziel. Ich habe einfach nicht erkannt, dass ich für den Anwendungsfall wichtig sein kann und doch fast nie bemerkt werde.

Von Kopf bis Fuß: Denken Sie sich doch einfach, dass der Anwendungsfall ohne Sie nicht vollständig wäre.

Alternativpfad: Ja. Genau das sagen mir 3.1 und 4.1 auch die ganze Zeit. Aber die sind natürlich Teil des Alternativpfads für den Fall, dass Kunden bereits ein Konto im System haben, und werden deswegen ständig verwendet. Da sagt man so was leicht!

Von Kopf bis Fuß: Lassen Sie sich nicht unterkriegen, Alternativpfad. Wir wissen, dass Sie ein wichtiger Teil des Anwendungsfalls sind!

Spitzen Sie Ihren Bleistift

Wie viele Szenarien gibt es in Tims und Ginas Anwendungsfall?

Auf wie viele unterschiedliche Weisen können Sie Tims und Ginas Anwendungsfall durchlaufen? Denken Sie daran, dass Sie manchmal einen der vielen Alternativpfade nehmen müssen und manchmal einen Alternativpfad vollständig überspringen können.

Tims und Ginas Hundetür, Version 2.3
Was die Tür macht

Hauptpfad

1. Fido bellt, damit man ihn rauslässt.
2. Die Bellerkennung »hört« ein Bellen.
3. Die Bellerkennung sendet der Tür eine Öffnen-Anfrage.
4. Die Hundetür geht auf.
5. Fido geht nach draußen.
6. Fido erledigt sein Geschäft.
   - 6.1. Die Tür schließt automatisch.
   - 6.2. Fido bellt, damit er wieder reingelassen wird.
   - 6.3. Die Bellerkennung »hört« wieder ein Bellen.
   - 6.4. Die Bellerkennung sendet der Tür eine Öffnen-Anfrage.
   - 6.5. Die Hundetür geht wieder auf.
7. Fido kommt wieder rein.
8. Die Tür schließt automatisch.

Alternativpfade

- 2.1. Tim oder Gina hört Fidos Bellen.
- 3.1. Tim oder Gina drückt auf den Fernsteuerungsknopf.
- 6.3.1. Tim oder Gina hört Fidos erneutes Bellen.
- 6.4.1. Tim oder Gina drückt auf den Fernsteuerungsknopf.

Sehen Sie sich unsere Lösungen auf Seite 128 an

Um Ihnen auf die Sprünge zu helfen, haben wir die Schritte aufgeschrieben, denen wir in dem Szenario oben gefolgt sind.

1. 1, 2.1, 3.1, 4, 5, 6, 6.1, 6.2, 6.3.1, 6.4.1, 6.5, 7, 8
2. _____
3. _____
4. _____
5. _____
6. _____
7. _____
8. _____

Eventuell brauchen Sie nicht alle Zeilen.

Spitzen Sie Ihren Bleistift

Lösung

Wie viele Szenarien gibt es in Tims und Ginas Anwendungsfall?

Auf wie viele unterschiedliche Weisen können Sie Tims und Ginas Anwendungsfall durchlaufen? Denken Sie daran, dass Sie manchmal einen der vielen Alternativpfade nehmen müssen und manchmal einen Alternativpfad vollständig überspringen können.

Tims und Ginas Hundetür, Version 2.3
Was die Tür macht

Hauptpfad	Alternativpfade
1. Fido bellt, damit man ihn rauslässt.	
2. Die Bellerkennung »hört« ein Bellen.	2.1. Tim oder Gina hört Fidos Bellen.
3. Die Bellerkennung sendet der Tür eine Öffnen-Anfrage.	3.1. Tim oder Gina drückt auf den Fernsteuerungsknopf.
4. Die Hundetür geht auf.	
5. Fido geht nach draußen.	
6. Fido erledigt sein Geschäft.	
6.1. Die Tür schließt automatisch.	
6.2. Fido bellt, damit er wieder reingelassen wird.	
6.3. Die Bellerkennung »hört« wieder ein Bellen.	6.3.1. Tim oder Gina hört Fidos erneutes Bellen.
6.4. Die Bellerkennung sendet der Tür eine Öffnen-Anfrage.	6.4.1. Tim oder Gina drückt auf den Fernsteuerungsknopf.
6.5. Die Hundetür geht wieder auf.	
7. Fido kommt wieder rein.	
8. Die Tür schließt automatisch.	

Das ist der Hauptpfad des Anwendungsfalls.

1. 1, 2.1, 3.1, 4, 5, 6, 6.1, 6.2, 6.3.1, 6.4.1, 6.5, 7, 8

Die beiden nehmen den Alternativpfad, bei dem Fido ausgesperrt wird.

2. 1, 2, 3, 4, 5, 6, 7, 8
3. 1, 2..1, 3.1, 4, 5, 6, 7, 8
4. 1, 2.1, 3.1, 4, 5, 6, 6.1, 6.2, 6.3, 6.4, 6.5, 7, 8

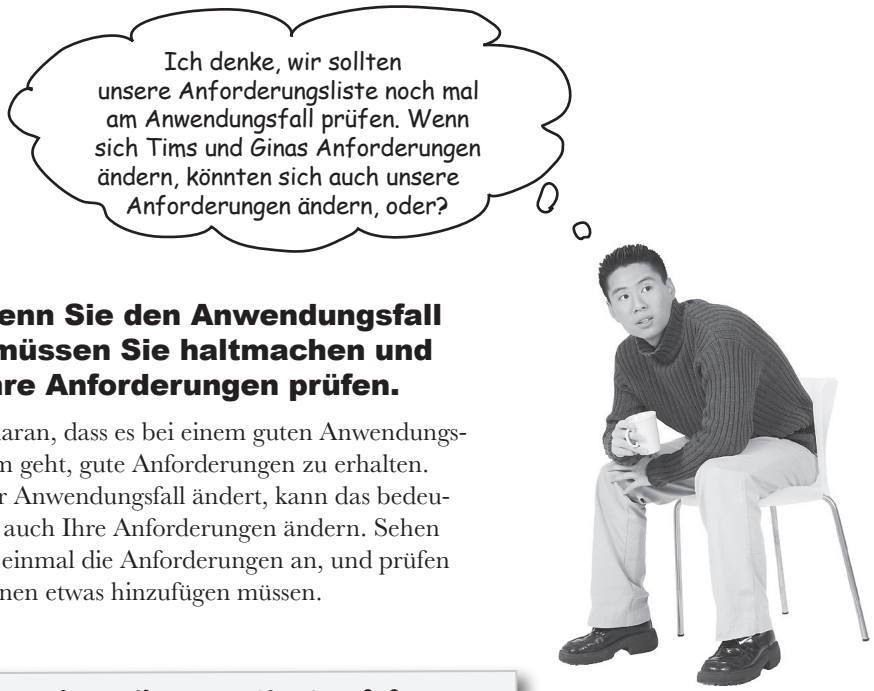
Wenn Sie Schritt 2.1 nehmen, nehmen Sie immer auch Schritt 3.1.

Wenn Sie Schritt 6.3.1 nehmen, nehmen Sie immer auch Schritt 6.4.1.

5. 1, 2, 3, 4, 5, 6, 6.1, 6.2, 6.3.1, 6.4.1, 6.5, 7, 8
6. 1, 2.1, 3.1, 4, 5, 6, 6.1, 6.2, 6.3.1, 6.4.1, 6.5, 7, 8
7. 1, 2, 3, 4, 5, 6, 6.1, 6.2, 6.3, 6.4, 6.5, 7, 8
8. <sonst nichts>

Machen Sie sich programmierbereit ...

Jetzt, da unser Anwendungsfall abgeschlossen ist und wir alle möglichen Szenarien für die Verwendung der Hundetür herausgefunden haben, sind wir bereit, Tims und Ginas neue Anforderungen zu bewältigen. Schaun wir mal, was wir tun müssen ...



Immer wenn Sie den Anwendungsfall ändern, müssen Sie haltmachen und erneut Ihre Anforderungen prüfen.

Denken Sie daran, dass es bei einem guten Anwendungsfall nur darum geht, gute Anforderungen zu erhalten. Wenn sich Ihr Anwendungsfall ändert, kann das bedeuten, dass sich auch Ihre Anforderungen ändern. Sehen wir uns noch einmal die Anforderungen an, und prüfen wir, ob wir ihnen etwas hinzufügen müssen.

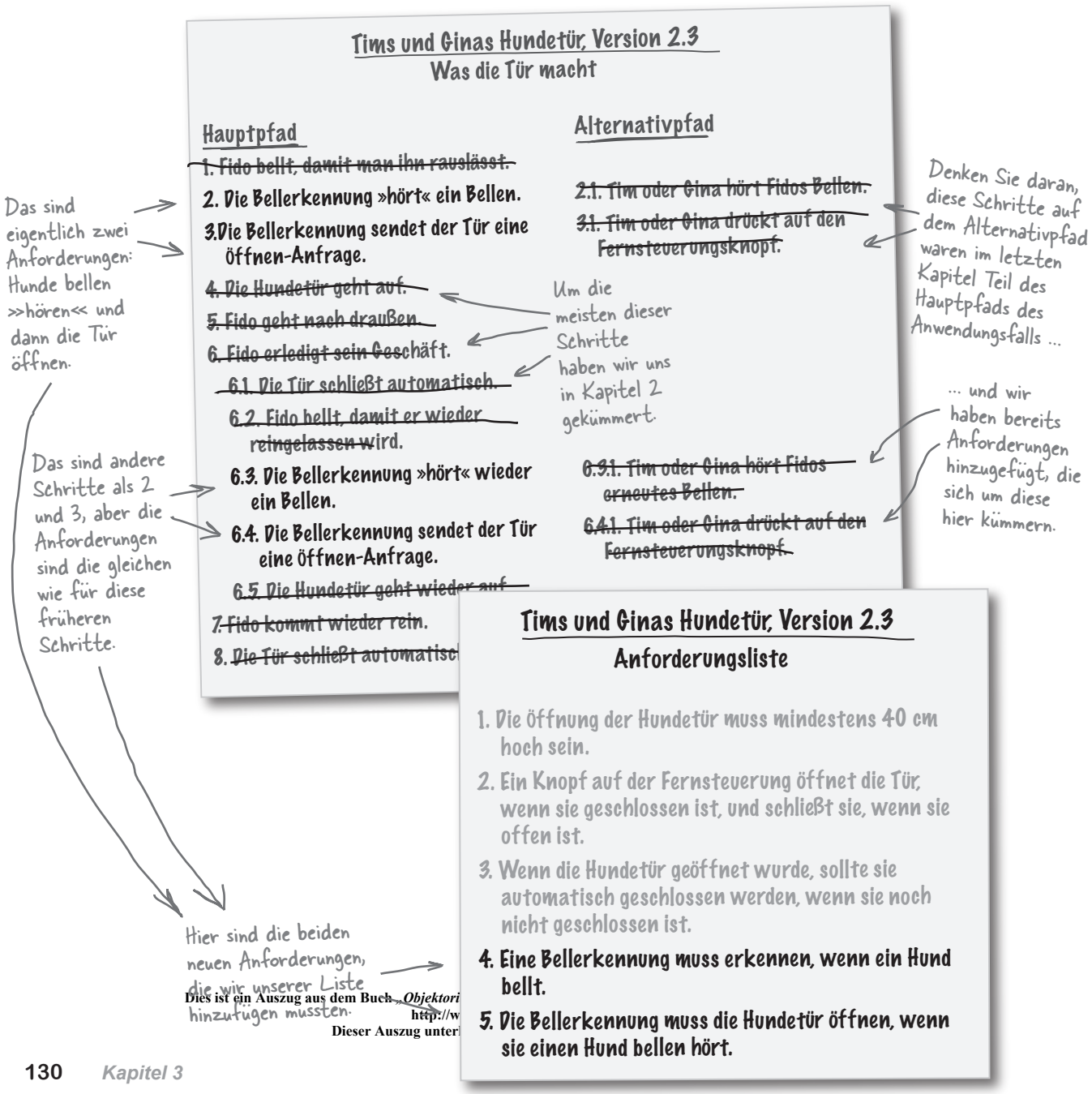
Fahren Sie fort und schreiben Sie alle zusätzlichen Anforderungen auf, die Ihnen aufgefallen sind, als Sie die Szenarien für die neue Hundetür auf Seite 128 durchgearbeitet haben.

Tims und Ginas Hundetür, Version 2.2
Anforderungsliste

1. Die Öffnung der Hundetür muss mindestens 40 cm hoch sein.
2. Ein Knopf auf der Fernsteuerung öffnet die Tür, wenn sie geschlossen ist, und schließt sie, wenn sie offen ist.
3. Wenn die Hundetür geöffnet wurde, sollte sie automatisch geschlossen werden, wenn sie noch nicht geschlossen ist.

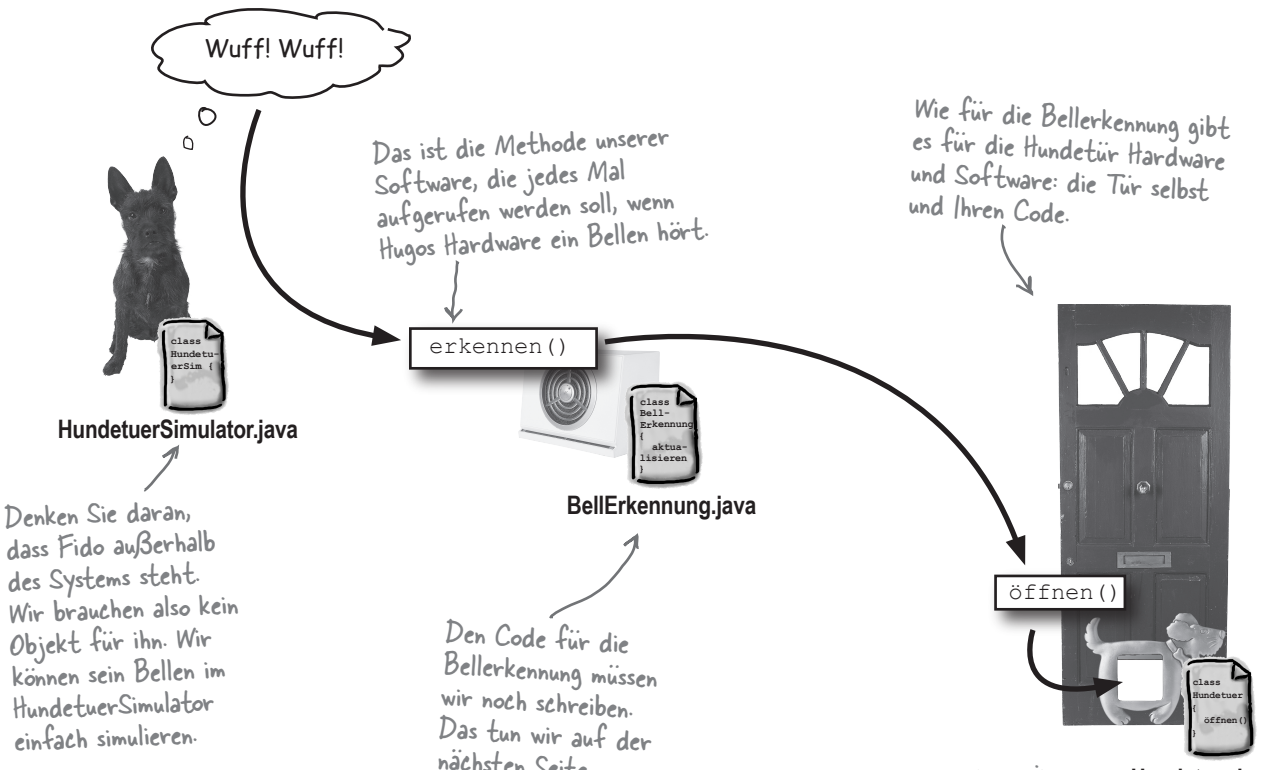
Die Anforderungsliste aufmöbeln

Wir müssen also zwei neue Alternativpfade behandeln, indem wir unserer Anforderungsliste ein paar zusätzliche Anforderungen hinzufügen. Wir haben etwas vorgearbeitet und die Schritte durchgestrichen, die unsere Anforderungsliste bereits erledigt. Nun sieht es so aus, als bräuchte unsere Anforderungsliste ein paar neue Ergänzungen:



Jetzt können wir wieder mit der Programmierung der Tür beginnen

Mit neuen Anforderungen kommt neuer Code. Wir brauchen etwas Bellen, eine Bellerkennung, die auf ein Bellen lauscht, und eine zu öffnende Hundetür:



Hab ich da ein »wuff« gehört?

Wir brauchen eine Software, die läuft, wenn Hugos Hardware ein Bellen »hört«. Erzeugen wir eine **`BellErkennung`**-Klasse und schreiben wir eine Methode, die wir verwenden können, um auf Bellen zu reagieren:

BellErkennung.java

```
public class BellErkennung {

  private Hundetuer tür;

  public BellErkennung(Hundetuer tür) {
    this.tür = tür;
  }

  public void erkennen(String bellen) {
    System.out.println("   BellErkennung: '" +
        bellen + "' gehört");
    tür.öffnen();
  }
}
```

Wir speichern die Hundetür, an die diese Bellerkennung geknüpft ist, in dieser Member-Variablen.

Die BellErkennung muss wissen, welche Tür sie öffnet.

Immer wenn die Hardware ein Bellen hört, ruft sie diese Methode mit dem Bellgeräusch auf, das sie gehört hat.

Wir müssen nur eine Meldung ausgeben, die dem System mitteilt, dass wir ein Bellen gehört haben ...

... und dann die Hundetür öffnen.

Es gibt keine Dummen Fragen

F: **Das war's? Sieht aber ganz so aus, als müsste die BellErkennung nicht sonderlich viel machen.**

A: Im Augenblick nicht. Da die Anforderungen einfach sind – öffne die Tür, wenn du ein Bellen hörst –, ist auch der Code ziemlich einfach. Immer wenn die Hardware ein Bellen hört, ruft sie **`erkennen()`** in unserer neuen Klasse **`BellErkennung`** auf, und wir öffnen die Hundetür. Denken Sie daran, die Dinge so einfach wie möglich zu lassen. Es gibt keinen Grund, Komplexität hinzuzufügen, wenn Sie sie nicht brauchen.

F: **Aber was passiert, wenn ein *anderer* Hund als Fido bellt? Sollte die Bellerkennung nicht sicherstellen, dass es Fido ist, der da bellt, bevor sie die Tür öffnet?**

A: Sehr interessante Frage! Die **`BellErkennung`** hört *jedes* Bellen, aber eigentlich wollen wir die Tür nicht für jeden *beliebigen* Hund öffnen, stimmt's? Später müssen wir eventuell zu diesem Punkt zurückkehren und das reparieren. Vielleicht sollten Sie darüber noch etwas nachdenken, während wir alles testen.

Ich denke, dass wir mit dieser neuen Klasse alles haben, was wir brauchen. Testen wir die BellErkennung und sehn wir mal, ob wir Tim und Gina wieder glücklich machen können.

Prüfen wir zunächst, ob wir uns um die neuen Anforderungen für Tims und Ginas Hundetür gekümmert haben:

Das ist eine weitere Hardware-Anforderung für Hugo. Für den Augenblick können wir den Simulator verwenden, um ein Bellen zu erhalten und die Software zu testen, die wir geschrieben haben.

Tims und Ginas Hundetür, Version 2.3
Anforderungsliste

1. Die Öffnung der Hundetür muss mindestens 40 cm hoch sein.
2. Ein Knopf auf der Fernsteuerung öffnet die Tür, wenn sie geschlossen ist, und schließt sie, wenn sie offen ist.
3. Wenn die Hundetür geöffnet wurde, sollte sie automatisch geschlossen werden, wenn sie noch nicht geschlossen ist.
4. **Eine Bellerkennung muss erkennen, wenn ein Hund bellt.**
5. **Die Bellerkennung muss die Hundetür öffnen, wenn sie einen Hund bellen hört.**

Das ist der Code, den wir gerade geschrieben haben ... immer wenn die Bellerkennung ein Bellen hört, öffnet sie die Tür.

Hmmm ... unsere Bellerkennung, >>erkennt<< doch nicht wirklich ein Bellen, oder? Sie öffnet die Tür auf ein BELIEBIGES Bellen. Zu diesem Punkt müssen wir später zurückkehren.

Die neue Hundetür einschalten

Anwendungsfälle, Anforderungen und Code haben alle zu diesem Punkt geführt. Schauen wir, ob alles funktioniert, wie es sollte.

HundetuerSimulator.java

❶ Aktualisieren Sie den HundetuerSimulator-Quellcode:

```
public class HundetuerSimulator {

  public static void main(String[] args) {
    Hundetuer tür = new Hundetuer();
    BellErkennung erkennung = new BellErkennung(tür);
    Fernsteuerung fs = new Fernsteuerung(tür);

    // Simulieren, dass die Hardware ein Bellen hört.
    System.out.println("Fido bellt.");
    erkennung.erkennen("Wuff");

    System.out.println("\nFido geht raus ...");

    System.out.println("\nFido ist fertig ...");

    try {
      Thread.currentThread().sleep(10000);
    } catch (InterruptedException e) { }

    System.out.println("... aber er ist ausgesperrt!");

    // Erneut simulieren, dass die Hardware ein Bellen hört.
    System.out.println("Fido bellt.");
    erkennung.erkennen("Wuff");

    System.out.println("\nFido ist wieder drinnen ...");
  }
}
```

Die BellErkennung erzeugen, mit der Tür verknüpfen und sie dann auf Bellen warten lassen.

Wir haben keine Hardware, simulieren also einfach, dass die Hardware ein Bellen hört.*

Hier tritt unsere neue BellErkennung-Software in Aktion.

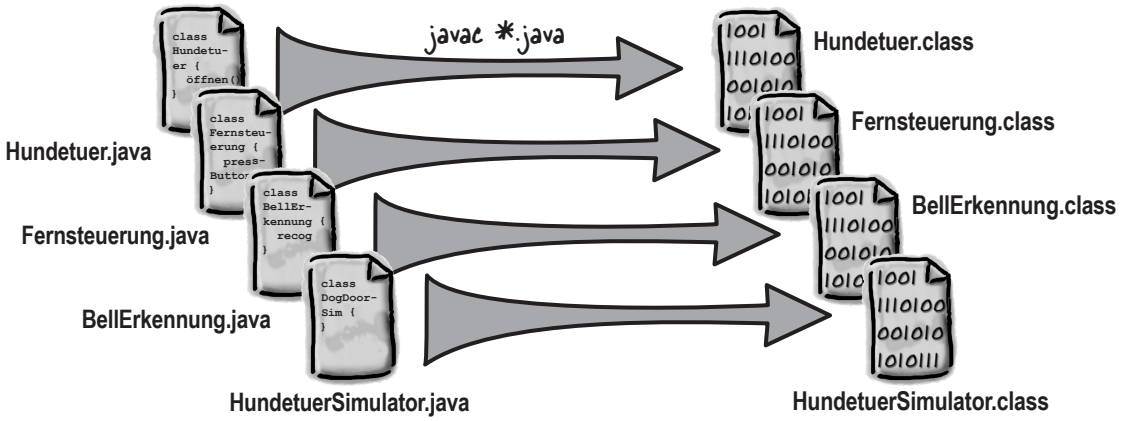
Hier simulieren wir, dass etwas Zeit verstreicht.

Wir testen den Vorgang, wenn Fido draußen ist, nur um sicherzustellen, dass alles funktioniert, wie es sollte.

Beachten Sie, dass dieses Mal Tim und Gina nicht einmal einen Knopf auf der Fernbedienung drücken müssen.

*Die Autoren dieses Buchs hatten die ernsthafte Absicht, Hardware dazuzupacken, die Hunde bellen hören kann ... aber die Marketing-Abteilung beharrte darauf, dass niemand ein Buch für 299,95 € kaufen würde. Können Sie sich das vorstellen?

❷ Kompilieren Sie Ihren gesamten Java-Quellcode erneut zu Klassen.



❸ Führen Sie den Code aus und beobachten Sie, wie die menschlose Hundetür in Aktion tritt.

```
Datei Bearbeiten Fenster Hilfe DuKläffstWieEinPudel
%java HundetuerSimulator
Fido bellt.
  BellErkennung: 'Wuff' gehört
Die Hundetür geht auf.

Fido geht raus ...

Fido ist fertig ...
... aber er ist ausgesperrt!
Fido bellt.
  BellErkennung: 'Wuff' gehört
Die Hundetür geht auf.

Fido ist wieder drinnen ...
```

Hier vergehen ein paar Sekunden, während Fido draußen spielt.

KOPF-NUSS

Es gibt ein großes Problem mit unserem Code, das sich im Simulator erkennen lässt. Können Sie herausfinden, was das Problem ist? Was würden Sie tun, um es zu reparieren?

Spitzen Sie Ihren Bleistift

Welches Szenario testen wir?

Können Sie herausfinden, welches Szenario des Anwendungsfalls wir testen? Schreiben Sie die Schritte auf, denen dieser Simulator folgt (blättern Sie zu Seite 123 zurück, um den Anwendungsfall erneut zu sehen).

Spitzen Sie Ihren Bleistift Lösung

Welches Szenario testen wir?

Haben Sie herausgefunden, welches Szenario des Anwendungsfalls wir testen? Hier sind die Schritte aus dem Anwendungsfall von Seite 123, denen wir gefolgt sind:

1, 2, 3, 4, 5, 6, 6.1, 6.2, 6.3, 6.4, 6.5, 7, 8

KOPF-NUSS

Es gibt ein großes Problem mit unserem Code, das sich im Simulator erkennen lässt. Können Sie herausfinden, was das Problem ist? Was würden Sie tun, um es zu reparieren?

Haben Sie herausgefunden, was das Problem mit der letzten Version unserer Hundetür ist?

In unserer neuen Version der Hundetür, schließt sich die Tür nicht mehr automatisch!

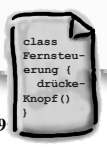
Das ist der Code, der in den Szenarien ausgeführt wird, in denen Tim und Gina den Knopf auf der Fernsteuerung drücken:

```
public void drückeKnopf() {
  System.out.println("Fernsteuerungsknopf gedrückt ...");
  if (tür.isOffen()) {
    tür.schließen();
  } else {
    tür.öffnen();

    final Timer timer = new Timer();
    timer.schedule(new TimerTask() {
      public void run() {
        tür.schließen();
        timer.cancel();
      }
    }, 5000);
  }
}
```

Wenn Tim und Gina auf den Knopf auf der Fernsteuerung drücken, richtet dieser Code auch einen Timer ein, um die Tür automatisch zu schließen.

Erinnern Sie sich? Dieser Timer wartet 5 Sekunden und fordert die Hundetür dann auf, sich selbst zu schließen.



Fernsteuerung.java

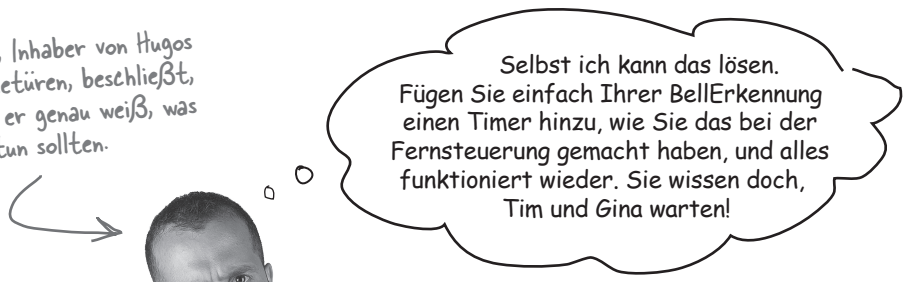
Aber in **`BellErkennung`** öffnen wir die Tür und schließen sie anschließend nicht mehr:

```
public void erkennen(String bellen) {
  System.out.println("   BellErkennung: " +
    "'" + bellen + "' gehört");
  tür.öffnen();
}
```

Wir öffnen die Tür, schließen sie aber nie.

BellErkennung.java

Hugo, Inhaber von Hugos Hundetüren, beschließt, dass er genau weiß, was Sie tun sollten.



Was halten SIE von Hugos Idee?

Hugo hat keine Ahnung. Ich will doch nicht den gleichen Code in die Fernsteuerung **und** in die Bellerkennung stecken.

Doppelter Code ist eine schlechte Idee. Aber wohin sollte der Code, der die Tür schließt?

Das Schließen der Tür ist doch eigentlich etwas, das die **Tür** tun sollte, nicht die Fernsteuerung oder die BellErkennung. Warum lassen wir die Hundetür sich nicht selbst schließen?

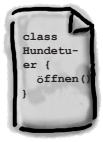
Machen wir es so, dass sich die Tür immer automatisch schließt.

Da Gina unter keinen Umständen will, dass die Tür offen bleibt, sollte sich die Tür *immer* automatisch schließen. Wir können den Code, der die Tür automatisch schließt, also in die Klasse **`Hundetuer`** verschieben. Dann schließt sich die Tür selbst, egal *wodurch* sie geöffnet wurde.

Obwohl das eine Entwurfsentscheidung ist, ist es ein Teil davon, die Software dazu zu bringen, dass sie funktioniert, wie der Kunde es möchte. Denken Sie daran, dass es in Ordnung ist, ein gutes Design zu verwenden, während man an der Funktionalität des Systems arbeitet.

Die Hundetür aktualisieren

Nehmen wir den Code zum Schließen der Tür aus der Klasse **Fernsteuerung** und stecken wir ihn in unseren **Hundetuer**-Code:



Hundetuer.java

Sie müssen auch die Importe für java.util.Timer und java.util.TimerTask hinzufügen.

```
public class Hundetuer {
  public void öffnen() {
    System.out.println("Die Hundetür geht auf.");
    offen = true;

    final Timer timer = new Timer();
    timer.schedule(new TimerTask() {
      public void run() {
        schließen();
        timer.cancel();
      }
    }, 5000);
  }

  public void schließen() {
    System.out.println("Die Hundetür geht zu.");
    offen = false;
  }
}
```

Das ist der gleiche Code wie zuvor in Fernsteuerung.java.

Jetzt schließt sich die Tür selbst ... auch wenn wir ein neues Gerät hinzufügen, das die Tür öffnen kann. Toll!

Die Fernsteuerung vereinfachen

Sie müssen diesen Code jetzt aus **Fernsteuerung** herausnehmen, da die Hundetür das automatische Schließen nun selbst erledigt:

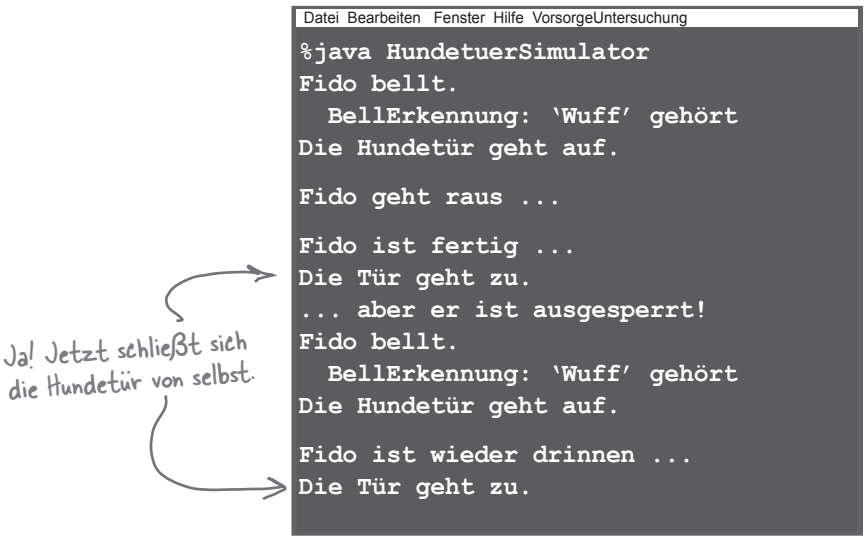
```
public void drückeKnopf() {
  System.out.println("Fernsteuerungsknopf gedrückt ...");
  if (tür.isOffen()) {
    tür.schließen();
  } else {
    tür.öffnen();

    final Timer timer = new Timer();
    timer.schedule(new TimerTask() {
      public void run() {
        tür.schließen();
        timer.cancel();
      }
    }, 5000);
  }
}
```

Fernsteuerung.java

Ein letzter Testlauf

Sie haben viele Änderungen an Tims und Ginas Hundetür vorgenommen, seit sie sich das erste Mal bei Ihnen gemeldet haben. Testen wir das, um zu prüfen, ob alles funktioniert. Nehmen Sie die Änderungen an **`Fernsteuerung.java`** und **`Hundetuer.java`** vor, damit sich die Tür selbst schließt, kompilieren Sie all Ihre Klassen und starten Sie erneut den Simulator:



KOPF-NUSS

Was würde passieren, wenn Tim und Gina entschieden, dass die Hundetür länger aufbleiben soll? Oder, dass sie sich schneller schließen soll? Fällt Ihnen eine Möglichkeit ein, Hundetuer so zu ändern, dass die Zeit, die verstreicht, bevor die Tür automatisch schließt, vom Kunden gesetzt werden kann?

Manchmal zeigen Änderungen der Anforderungen Probleme mit einem System auf, deren Existenz Ihnen zuvor nicht klar war.

Veränderung ist konstant, und Ihr System sollte jedes Mal, wenn Sie daran arbeiten, besser werden.

Spitzen Sie Ihren Bleistift

Schreiben Sie Ihr eigenes Entwurfsprinzip!

Sie haben in diesem Kapitel in Bezug auf doppelter Code und das automatische Schließen der Tür ein wichtiges Entwurfsprinzip verwendet. Versuchen Sie, das Entwurfsprinzip zusammenzufassen, das Sie Ihrer Meinung nach gelernt haben:

Entwurfsprinzip

_____

_____

_____

In diesem Kapitel werden Sie keine Antwort auf dieses Rätsel finden. Aber etwas später werden wir noch einmal darauf zurückkommen. Geben Sie sich trotzdem Mühe!

Ihr erweiterter OOA&D-Werkzeugkasten

In diesem Kapitel haben Sie eine Menge gelernt, und jetzt ist es Zeit, das, was Sie gelernt haben, in Ihren OOA&D-Werkzeugkasten zu packen. Sehen Sie sich auf dieser Seite noch einmal an, was Sie gelernt haben, und wenden Sie es dann im OOA&D-Kreuzworträtsel auf der nächsten Seite an.

Anforderungen

Gute Anforderungen stellen sicher, dass Ihr System funktioniert, wie es Ihre Kunden erwarten.

Achten Sie darauf, dass Ihre Anforderungen alle Schritte in den Anwendungsfällen für Ihr System abdecken.

Nutzen Sie Anwendungsfälle, um die Dinge herauszufinden, die Ihre Kunden Ihnen zu sagen vergessen haben.

Ihre Anwendungsfälle legen unvollständige oder fehlende Anforderungen offen, die Sie Ihrem System eventuell hinzufügen müssen.

Mit der Zeit werden sich Ihre Anforderungen immer ändern (und ausdehnen).

Sie haben nur ein neues Anforderungs-prinzip gelernt, aber das ist wichtig!

OO-Prinzipien

Kapseln Sie, was veränderlich ist.

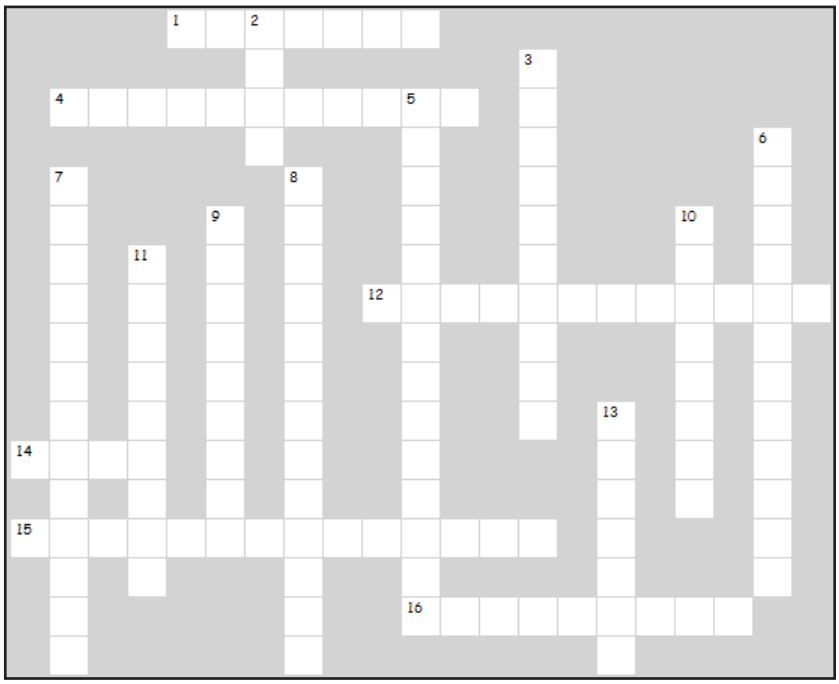
Kapselung hat uns geholfen zu erkennen, dass sich die Hundetür selbst um das Schließen kümmern sollte. Wir haben das Verhalten der Tür vom restlichen Code der Anwendung getrennt.

Punkt für Punkt

- Während ein Projekt vorangeht, **ändern** sich immer die Anforderungen.
- Wenn sich Anforderungen ändern, muss sich das System weiterentwickeln, um die neuen Anforderungen zu bewältigen.
- Wenn Ihr System auf neue oder andere Weise arbeiten muss, beginnen Sie mit der Aktualisierung Ihres Anwendungsfalls.
- Ein **Szenario** ist ein bestimmter Pfad durch einen Anwendungsfall, vom Anfang bis zum Ende.
- Ein einziger Anwendungsfall kann mehrere Szenarien haben, wenn alle Szenarien das gleiche Kundenziel haben.
- **Alternativpfade** können Schritte sein, die nur gelegentlich eintreten, oder vollständig andere Pfade durch Teile eines Anwendungsfalls bieten.
- Wenn ein Schritt für die Funktionsweise eines Systems optional ist oder ein Schritt einen Alternativpfad durch ein System bietet, verwenden Sie nummerierte Unterschritte wie 3.1, 4.1 und 5.1 oder 2.1.1, 2.2.1 und 2.3.1.
- Sie sollten fast immer versuchen, **doppelten Code zu vermeiden**. Er ist ein Wartungsalbtraum und weist in der Regel auf Probleme im Design eines Systems hin.

OOA&D-Kreuzworträtsel

Das mit den Puzzeln geht weiter. Prüfen Sie, dass Sie alle Schlüsselkonzepte in diesem Kapitel verstanden haben, indem Sie dieses Kreuzworträtsel durcharbeiten. Alle Antwortwörter finden sich irgendwo in diesem Kapitel.



WAAGERECHT

1 Anwendungsfälle haben meist _____ Szenarien.
4 So wird der Hauptpfad auch genannt.
12 Sollten Sie in Ihrem Anwendungsfall verwenden, wenn ein Schritt optional ist.
14 Mit ihr ändern sich fast immer auch die Anforderungen.
15 Sollten Sie aktualisieren, wenn sich Ihr System ändert, bevor Sie Code schreiben.
16 Vermeiden Sie grundsätzlich _____ Code.

SENKRECHT

2 Wollte den Timer doppelt einbauen.
3 Die einzige Konstante bei der Analyse und dem Design von Software.
5 Eine Gruppe von Schritten, die in Ihrem Anwendungsfall nicht immer eintreten.
6 Das teilen alle Szenarien in einem Anwendungsfall.
7 Mussten wir unserer Hundetür hinzufügen, um Tim und Gina glücklich zu machen.
8 Ändern sich in der Regel auch, wenn sich der Anwendungsfall ändert.
9 Dem folgen Sie in einem Anwendungsfall meistens.
10 Viele richtige Anwendungen schließen oft Software und _____ ein.
11 Haben wir für das Schließen der Hundetür verantwortlich gemacht.
13 Machen Sie das mit Dingen, die sich ändern.

				M	E	H	R	E	R	E									
						U						V							
	E	R	F	O	L	G	S	P	F	A	D		E						
						O				L			R					B	
	B						A			T			N					E	
	E				H		N			E			D				H		N	
	L		H		A		F			R			E				A		U	
	L		U		U		O		U	N	T	E	R	S	C	H	R	I	T	T
	E		N		P		R			A			U				D		Z	
	R		D		T		D			T			N				W		E	
	K		E		P		E			I			G		K		A		R	
Z	E	I	T		F		R			V					A		R		Z	
	N		U		A		U			P					P		E		I	
A	N	W	E	N	D	U	N	G	S	F	A	L	L		S				E	
	U		R				G			A					E				L	
	N						E			D	O	P	P	E	L	T	E	N		
	G						N								N					