Planung mit **Featurelisten**

Nach dem Überblick in der letzten Ausgabe [1] wird Ihnen nun der Featurebegriff aus dem Feature Driven Development (FDD) vorgestellt. Es wird gezeigt, wie Features definiert und wie entlang von Featurelisten geplant wird. Dabei stellen wir heraus, welche FDD-Techniken auch in anderen Projekten nutzbringend verwendet werden können.

von Stefan Roock, Henning Wolf

Nach der Erstellung des Überblickmodells mit Modeling in Color [2] wird in Feature Driven Development die Featureliste erstellt. Dafür gibt es eine übersichtliche Strukturierung und ein Benennungsschema als Anleitung. Auf dieser Basis erfolgt die Planung und feste Zuordnung von Entwicklern zu Klassen des Übersichtsmodells. Es geht hier also um die Prozesse #2 („Erstelle eine Featureliste“) und #3 („Plane je Feature“) des Feature Driven Development (Abb. 1).

Das Featureschema

Features werden nach einem einheitlichen Schema beschrieben: <*Aktion*> <*Ergebnis*> <*Objekt*>. Im Beispiel „Berechne Gesamtsumme der Verkäufe“ ist „Berechne“ die Aktion, „Gesamtsumme“ das Ergebnis und „Verkäufe“ das Objekt für die Aktion. Wichtig ist, dass die Features für den Kunden und die Fachexperten verständlich bleiben müssen. Außerdem muss jedes Feature aus Kundensicht einen Fortschritt bedeuten. „Öffne Transaktion auf der Datenbank.“ ist demnach in den meisten Projekten kein gültiges Feature.

Die Benennung der Aktion in Befehlsform („Berechne“) ist ungewöhnlich, aber sehr wertvoll. In vielen Projekten werden Anforderungen so benannt, dass die Verben verschleiert oder ganz weggelassen werden („Artikelverwaltung“). Dadurch muss man in der Regel nicht nur die Überschrift einer Anforderung lesen, sondern auch noch den Beschreibungstext, um zu verstehen, was genau gemeint ist. Häufig fällt es Entwicklern bei so aufgeschriebenen Anforderungen schwer, den Aufwand abzuschätzen. Oft entblößt sich die „versteckte“ Komplexität der Anforderung erst, wenn mit der Programmierung begonnen wird.

Wir haben die Erfahrung gemacht, dass das Feature-Beschreibungsschema diese Probleme sehr effektiv adressiert. Zum einen reicht bei der Featureformulierung meist tatsächlich nur die Benennung in der Form <Aktion> <Ergebnis> <Objekt>, ohne dass man einen weiteren Beschreibungstext benötigt. Bei Gesprächen mit dem Kunden braucht man dann auch nur die Übersichtsliste der Features, während man bei der „verbfreien“ Form der Anforderungen zusätzlich die Detailbeschreibungen der Anforderungen benötigt. Zusätzlich wird allen Beteiligten mit der Featureform viel klarer, was sich alles hinter einem Anforderungspaket verbirgt. Man wird viel seltener von versteckter Komplexität überrascht.

Ein Gegenbeispiel belegt dies: Würden wir ein Feature wie „Verwalte Fahrzeuge“ akzeptieren, so wäre unklar, was sich hinter dem Verwalten alles verbirgt. Brauchen wir eine Auflistung aller Fahrzeuge? Brauchen wir eine Suche über alle Fahrzeugeigenschaften? Welche Statistiken und Ausdrucke werden benötigt? Müssen alte Fahrzeuge archiviert werden? Muss die Tabelle der Auflistung sortierbar sein? Gibt es ein Rechtekonzept oder darf jeder Anwender alles mit der Fahrzeugverwaltung tun? Eine Auflistung der folgenden Features hingegen sorgt für deutlich mehr Klarheit:

- Erstelle leere Fahrzeugliste.
- Füge Fahrzeug der Fahrzeugliste hinzu.
- Ermittle Fahrzeugliste an.
- Lösche Fahrzeug.
- Wähle Fahrzeug aus Fahrzeugliste aus.
- Ändere Fahrzeugeigenschaften eines Fahrzeugs.
- Sortiere Fahrzeugliste.

Navigator FDD-Artikelserie

Teil 1: FDD-Überblick
Teil 2: Planung mit Featurelisten
Teil 3: Entwurf und Konstruktion der Features
Teil 4: FDD-Projektmanagement

Was hier nicht als Feature benannt ist, wird auch nicht umgesetzt, also für dieses Beispiel kein Rechtekonzept, keine Ausdrucke und keine Suche.

Definition der Featureliste

Die Featureliste wird von den Chefprogrammierern auf Basis des Gesamtmodells (Prozess #1) erstellt und anschließend mit den Fachexperten abgestimmt. In der Featureliste sind alle Features vorhanden. Um den Überblick zu behalten, werden die Features hierarchisch gruppiert: Features werden in Geschäftstätigkeiten (engl. *Business Activities*) gruppiert und Geschäftstätigkeiten wiederum in Fachgebiete (engl. *Subject Areas*). Abbildung 2 visualisiert das Hierarchisierungsprinzip. Der Prozess wird verlassen, wenn alle Anforderungen als Features auf der Featureliste definiert und mit den Fachexperten abgestimmt sind.

Durch die Abstimmung mit den Fachexperten wird der Kunde bzgl. der Featurelisten mit ins Boot geholt. Fehlen in der Featureliste einzelne Features, steht der Kunde mit in der Verantwortung. In diesem Sinne hat die Featureliste nicht nur die Funktion, feingranulare Arbeitspakte für die Entwickler zu definieren, sondern auch den Funktionsumfang des Systems zu definieren.

Mitunter werden bei der Definition der Featureliste „künstliche" Features für die Benutzungsoberfläche (UI-Features) und die Infrastruktur geschaffen. Insbesondere UI-Features bringen häufig zusätzliche Klarheit, weil sie Anforderungen an die Benutzungsoberfläche explizit machen und häufig keine 1:1-Beziehung zwischen fachlichen und UI-Features existiert. Die fachlichen Features „Erstelle leere Fahrzeugliste" und „Füge Fahrzeug der Fahrzeugliste hinzu" korrespondieren beispielsweise mit den UI-Features „UI: Zeige Fahrzeugliste", „UI: Zeige Button für leere Fahrzeugliste" und „UI: Zeige Hinzufügen-Button für Fahrzeuge".

Plane je Feature

Im dritten FDD-Prozess planen der Projektleiter, der Entwicklungsleiter und die Chefprogrammierer die Reihenfolge, in der Features realisiert werden sollen. Dabei richten sie sich nach den Abhängigkeiten zwischen den Features (bzw. den Geschäftstätigkeiten), der Auslastung der Programmierteams sowie der Komplexität der Features.

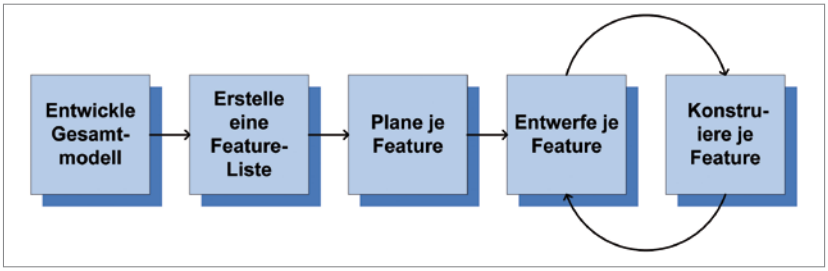

Abb. 1: Die fünf FDD-Prozesse

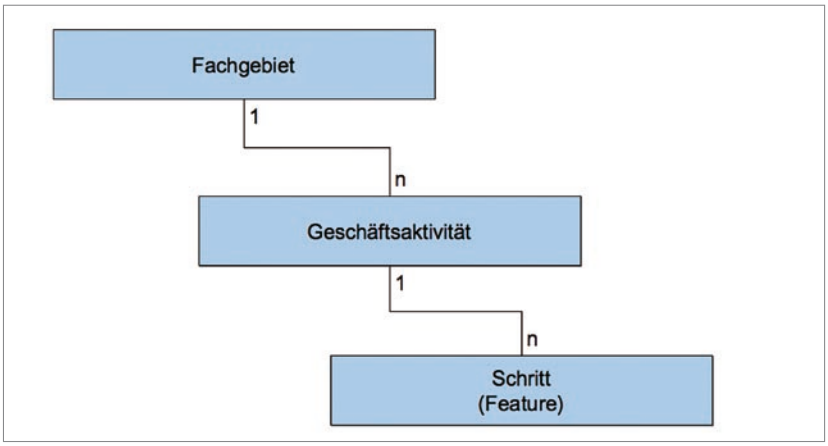

Abb. 2: Anforderungsorganisation in FDD

Auf Basis des Plans werden die Fertigstellungstermine je Geschäftstätigkeit festgelegt. Jede Geschäftstätigkeit bekommt einen Chefprogrammierer als Besitzer zugeordnet. Außerdem werden für die bekannten Kernklassen Entwickler als Besitzer festgelegt (engl. *Class Owner List*). Auf dieser Basis werden Featurepakete geschnürt und durch temporäre Featureteams realisiert. Bei der Featureplanung ist es essenziell, für die reibungslose Abwicklung der Featureentwicklung zu sorgen. Zum einen müssen Flaschenhälse vermieden werden. Kein Entwickler soll lange auf Zuarbeiten anderer Entwickler warten müssen. Gleichzeitig sollen alle Entwickler möglichst ausgelastet werden.

Für jedes Feature wird festgelegt, welche Klassen bei der Erledigung des Features beteiligt sein müssen. Dies lässt sich dank der Feingranularität der Features und des Feature-Beschreibungsschemas meist recht einfach bewerkstelligen. Sowohl <Ergebnis> als auch <Objekt> sind meist erste Kandidaten für beteiligte Klassen. Es kommen aber ggf. auch zusätzliche Service- oder Oberflächenklassen hinzu. Die Zuordnungsliste „Klasse zu Feature" findet in die Planung Eingang, indem sie eine wesentliche Grundlage für die Festlegung der Class-Ownership auf einzelne Entwickler ist. Allerdings kann es auch in FDD-Projekten dazu kommen, dass man später im Projektverlauf von der ursprünglichen Planung abweicht bzw. diese überarbeitet. Das würde der Projektleiter zum Beispiel auch bezüglich der Class-Ownership tun, wenn ein Entwickler häufig überlastet ist. Er sollte dann einen Teil „seiner" Klassen einem anderen Entwickler übergeben.

Alle Class-Owner der beteiligten Klassen zu einem Feature bilden das so genannte Featureteam. Diese Teams bilden sich dynamisch je Feature und arbeiten gemeinsam ein Feature ab (mehr dazu im nächsten Teil dieser Serie). Abbildung 3 zeigt für die Prozesse #2 und #3 die einzelnen Aktivitäten im Überblick.

Featureaufwände

Ist die Featureliste erstellt, lässt sich eine Aufwandsschätzung vornehmen. Dabei wird jedes Feature im Aufwand geschätzt, und die Einzelaufwände werden zur Gesamtschätzung aufaddiert.

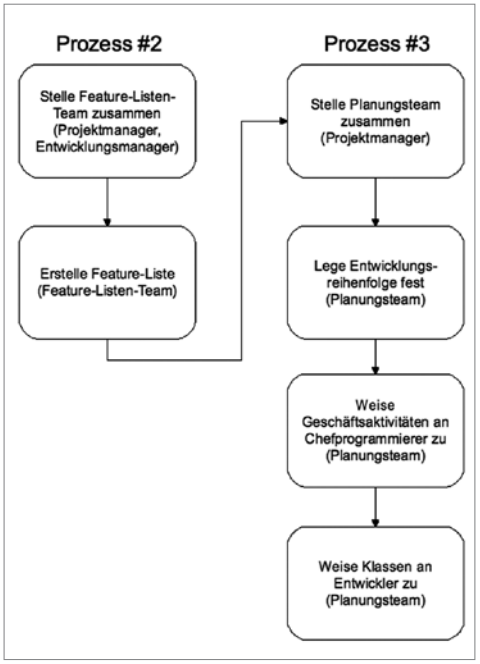

Abb. 3: Aktivitäten der Prozesse #2 und #3

FDD ist für Festpreiskonstellationen gut geeignet, denn die pragmatische Entwicklung des Gesamtmodells und die Erstellung der detaillierten Featureliste im Vorweg liefern einen angenehm konkreten Festpreisrahmen.

Für die konkrete Aufwandsschätzung der einzelnen Features liefert FDD keine konkrete Hilfestellung. Es kommen hier also übliche Schätzverfahren in Personentagen, mit Aufwands- oder Story-Punkten zum Tragen [3]. Die einzelnen Features sind aber dank des Feature-Beschreibungsschemas meist angenehm klein und feingranular, was die Schätzung erleichtert. Als Obergrenze für den Aufwand eines Features legt FDD 14 Tage fest, was je nach Anzahl beteiligter Class-Owner durchaus bis zu 50 PT werden können. Sollte sich beim Schätzen herausstellen, dass ein Feature aufwändiger als 14 Tage ist, muss es in mehrere Einzelfeatures aufgeteilt werden. Typischerweise sind einzelne Features aber viel geringer im Aufwand als 14 Tage, meist in der Größenordnung weniger Tage.

Der Gesamtaufwand gibt schließlich auch einen wichtigen Eindruck davon, wie viele Entwickler benötigt werden, um die Features in einem bestimmten Zeitrahmen abzuarbeiten. Dabei geht Feature Driven Development von maximal sechs Monate langen Projekten aus. Es empfiehlt sich, im Zweifelsfall größere Projekte in mehrere aufeinanderfolgende aufzuteilen, damit der Kunde in jedem Fall nach spätestens sechs Monaten erneut die Möglichkeit hat, auf die ausgewählten Features und ggf. auch die Modellierung Einfluss zu nehmen.

Aufwandsschätzung durch Featuremodellierung ersetzen

Wir haben gute Erfahrungen in Projekten mit der Maßgabe gemacht, dass jedes Feature innerhalb eines Personentages realisierbar sein soll. Größere Features müssen dann zerlegt werden. So erhält man eine so große Anzahl von Features, dass einzelne Aufwandsausreißer nicht so stark ins Gewicht fallen und man gar keine differenzierende Aufwandsschätzung mehr benötigt. Man zählt einfach die Anzahl der Features und multipliziert sie mit einem Durchschnittswert für den Aufwand (bei uns je nach Projekt 0,5 bis 1 Personentag pro Feature).

Damit ersetzt man mittels FDD die Aufwandsschätzung durch Featuremodellierung. Dieser Vorgang findet bei Aufwandsschätzungen klassischer Anforderungen häufig implizit statt, indem die Entwickler sich überlegen: „Für die Anforderung muss ich X machen und dann Y und dabei darf ich Z nicht vergessen." Diese impliziten Überlegungen gehen nach der Schätzung aber wieder verloren. Bei FDD werden solche Überlegungen im Featureschema kanalisiert und stehen als Featureliste sofort zur Verfügung.

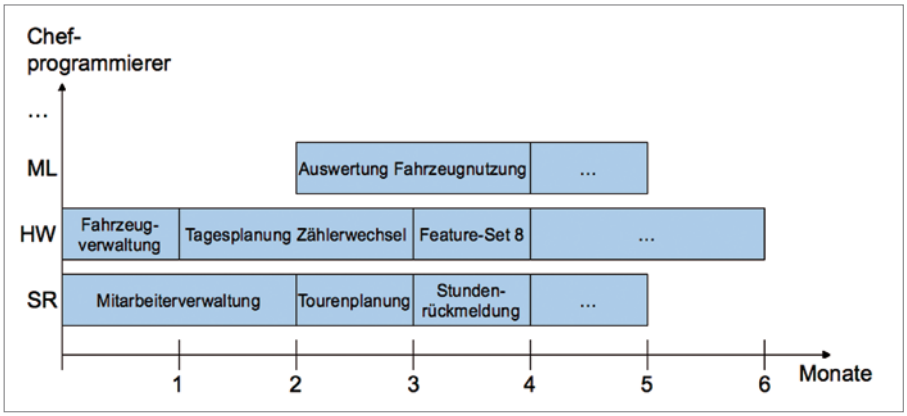

Abb. 4: Planung nach Geschäftstätigkeiten

Abarbeitungsreihenfolge festlegen

Features werden in Geschäftstätigkeiten organisiert. Die Geschäftstätigkeiten sind meist eine gute Einheit zur Planung der Priorisierung und damit Reihenfolge der Abarbeitung. Dabei werden je nach Teamgröße eine Menge von Geschäftstätigkeiten parallel abgearbeitet. Der Aufwand einer Geschäftstätigkeit lässt sich leicht über die Summierung der zugehörigen Features ermitteln. Damit lässt sich für jede Tätigkeit je nach möglicher Parallelisierbarkeit der enthaltenen Features angeben, in welchem Monat es voraussichtlich enden wird. So werden dann die zugehörigen Entwickler für die nächste Tätigkeit zu den entsprechenden Klassen verfügbar. Jeder Geschäftstätigkeit ist ein Chefprogrammierer zugeordnet, der für die gesamte Geschäftstätigkeit Verantwortung übernimmt. Bei der Priorisierung ist es eine gute und übliche Strategie, dass man mit besonders risikobehafteten Anteilen des Projekts beginnt, um die Risiken früh zu minimieren.

Bewertung

Die Definition von Features in dem strikten Schema <Aktion> <Ergebnis> <Objekt> führt zu prägnanten, einheitlich beschriebenen Anforderungen, die auch sehr ähnliche Umfänge aufweisen. Schwammig definierte Anforderungen werden unwahrscheinlicher. Wenn man in Meetings über Anforderungen spricht, reicht in der Regel die Übersichtsliste der Features. Bei klassisch definierten Anforderungen benötigt man meist die Übersichtsliste und alle Detailbeschreibungen. In diesem Fall verliert man schneller die Übersicht.

Der einfachste Mechanismus zur Aufwandsschätzung ist das simple Aus-

zählen von Features. Das Featureschema schafft einheitlich kleine Features, die diese Art der Aufwandsschätzung ermöglichen. Die Modellierung der Features im Team schafft ein gemeinsames Verständnis der Anforderungen, das häufig über das hinausgeht, was man mit klassischen Anforderungsbeschreibungen erreichen kann. Nicht zuletzt wird der Kunde durch die fachliche Beschreibung der Features und die Abstimmung mit Fachexperten mit ins Boot geholt. Man bekommt ein Commitment auf die zu entwickelnden Features und lässt weniger Spielraum, um später zusätzliche Anforderungen in die Features hinein zu interpretieren.

Übertragung

Anforderungen kann man auch bei anderem Vorgehen im FDD-Featureschema beschreiben. Mitunter fällt einem für das Schema kein Ergebnis oder kein Objekt ein. Dann kann man auch erstmal ohne leben. Die Aktion schafft den Großteil der Prägnanz. Wenn man nur das Featureschema übernimmt, kann es leicht passieren, dass der Kunde mit Anforderungen auf einer grobgranulareren Ebene hantiert. Man muss dann seine Anforderungen als zusätzliche Gruppierungsebene für die Features verwenden, damit er die Beziehung seiner Anforderungen zu den Features versteht.

Wenn man sich außerhalb klassischer Geschäftsanwendungen bewegt passt mitunter die Geschäftstätigkeit als Gruppierung von Features nicht mehr. Dann muss man sich einen anderen Gruppierungsmechanismus so schaffen, dass ein Feature immer nur in einer Gruppe vorkommt, die Gruppierung stabil gegen fachliche Änderungen ist und die Gruppe ihre Features idealerweise zeitlich gruppiert. In diesem Sinn passen in einer Internetanwendung die einzelnen Seiten nur bedingt. Die Seiten als Gruppierungsmechanismus sind relativ instabil gegen Änderungen der Anforderungen. Besser eignen sich die vom Benutzer gewünschten Aktionen (z.B. „Überweisung durchführen“).

In kleinen Teams kann der dritte Prozess (*Plane je Feature*) teilweise oder ganz entfallen. Wenn die Entwickler alle im selben Raum sitzen, können sie sich spontan und direkt darüber abstimmen, wer wann was macht. Je mehr Entwickler zu organisieren sind, desto wichtiger wird die Featureplanung.

Gute Erfahrungen haben wir damit gemacht, die Features als Zeilen auf Flipchart-Zettel zu schreiben. Die Flipcharts haben wir an die Wände im Teamraum gehängt. Immer wenn ein Feature erledigt war, wurde es durchgestrichen. Dadurch ist eine sehr hohe Transparenz über den Projektzustand und ggf. notwendige Abstimmungen entstanden. ■


Stefan Roock und **Henning Wolf** sind Berater bei der akquinet it-agile GmbH. Sie arbeiten als Entwickler, Projektleiter und agile Methodenberater. Sie haben seit 1998 Erfahrungen mit agilen Methoden. Über XP haben sie ein Erfahrungsbuch geschrieben, in Scrum und FDD sind sie zertifiziert.

Links & Literatur

[1] Stefan Roock, Henning Wolf: Feature Driven Development – Überblick, in: *Java Magazin*, 02.2008, S. 118–121

[2] Thilo Fotscher: Mit Color Modeling farbige Domänenmodelle erstellen, in: *Java Magazin*, 12.2007, S. 37–42

[3] Henning Wolf, Stefan Roock, Martin Lippert: eXtreme Programming, 2. Aufl., dpunkt, 2005

[4] FDD-Beschreibung auf 10 Seiten: www.nebulon.com/articles/fdd/download/fddprocessesA4.pdf

[5] Stephen R. Palmer, John M. Felsing: A Practical Guide to Feature Driven Development, Prentice Hall International, 2002

[4] www.netbeans.org/products/platform/

Impressum

Verlag:
Software & Support Verlag GmbH

Anschrift der Redaktion:
Java Magazin
Software & Support Verlag GmbH
Geleitsstraße 14
D-60599 Frankfurt am Main
Tel. +49 (0) 69 6300890
Fax. +49 (0) 69 63008989
redaktion@javamagazin.de
www.javamagazin.de

Chefredakteur: Sebastian Meyen
Redaktion: Claudia Schaumlöffel
Chefin vom Dienst: Nicole Bechtel
Schlussredaktion: Nicole Bechtel, Katharina Klassen, Frauke Pesch
Leitung Grafik & Produktion: Jens Mainz
Layout, Titel: Kristin Brockmann, Jessica Demirkaya, Melanie Hahn, David Heinze, Dominique Kalbassi, Jens Mainz, Michel Michiels-Corsten, Katharina Ochsenhirt, Maria Rudi
CD/DVD-Erstellung: Daniel Zuzek

Autoren dieser Ausgabe:
Manuel Aldana, Pavlo Baron, Carsten Eilers, Malte Geßner, Sven Haiges, Alexandra Imrie, Christian Kappert, Kristian Köhler, Jakob Külzer, Torsten Lunze, Markus Plesser, Dr. Stefan Queins, Anja Ranft, Bruce Sams, Eric Schmieders, Dietrich Schulten, Markus Stäuble, Marc Steger, Matthias Weßendorf, Henning Wolf, Dieter Wulz, Anton Zrzavy

Anzeigenverkauf:
Software & Support Verlag GmbH
Patrik Baumann
Tel. +49 (0) 69 6300890
Fax. +49 (0) 69 63008989
pbaumann@javamagazin.de

Es gilt die Anzeigenpreisliste Nummer 11

Pressevertrieb:
DPV Network
Tel.+49 (0) 40 378456261,
www.dpv-network.de

Druck: PVA Landau
ISSN: 1619-795X

Abo-Service:
Software & Support Verlag GmbH
Tel. +49 (0) 69 6300890
Fax +49 (0) 69 63008989
www.javamagazin.de/service/

Abonnementpreise der Zeitschrift:

Inland:	12 Ausgaben	€ 79,–
Europ. Ausland:	12 Ausgaben	€ 89,–
Studentenpreis (Inland)	12 Ausgaben	€ 69,–
Studentenpreis (Ausland):	12 Ausgaben	€ 79,–

Einzelverkaufspreis:

Deutschland:	€ 7,50
Österreich:	€ 8,60
Schweiz:	sFr 15,80

Erscheinungsweise: monatlich

© Software & Support Verlag GmbH

Alle Rechte, auch für Übersetzungen, sind vorbehalten. Reproduktionen jeglicher Art (Fotokopie, Nachdruck, Mikrofilm oder Erfassung auf elektronischen Datenträgern) nur mit schriftlicher Genehmigung des Verlages. Jegliche Software auf der Begleit-CD zum *Java Magazin* unterliegt den Bestimmungen des jeweiligen Herstellers. Eine Haftung für die Richtigkeit der Veröffentlichungen kann trotz Prüfung durch die Redaktion vom Herausgeber nicht übernommen werden. Honorierte Artikel gehen in das Verfügungsrecht des Verlags über. Mit der Übergabe der Manuskripte und Abbildungen an den Verlag erteilt der Verfasser dem Herausgeber das Exklusivitätsrecht zur Veröffentlichung. Für unverlangt eingeschickte Manuskripte, Fotos und Abbildungen keine Gewähr. Java™ ist ein eingetragenes Warenzeichen der Sun Microsystems Inc.

ivw

hosted by Host Europe