Inhalt

Feature

Ähnlich	Vertragsfeature, Releasefeature, Produktfeature, Iterationsfeature
Abgegrenzt	Feature Driven Development
Siehe auch	(Scrum-)Benutzergeschichte
Semantisches Netz	(Scrum-)Epos, Definition von Bereit zur Umsetzung, Feature Driven Development, Featureteam, Featureteam vs. Komponententeam, Planungsrelevantes Feature, Product Backlog, Product-Backlog-Item, Sprint Backlog, Story Points, Teilabnahme, Vorabnahme

Einleitung

Jeder von uns hat das Wort Feature schon einmal in dem einen anderen Kontext verwendet, spätestens beim Witzeln mit "it's not a bug, it's a feature". Und die meisten von uns hatten auch schon einmal eine Softwareverpackung in der Hand, auf dem Features die Leistung des enthaltenen Produktes anpreisen. Da steht dann zum Beispiel "Integration in ", "Silbentrennung", "CSS Editor", "komfortable Kundenverwaltung", "mehrplatzfähiger Gruppenkalender", "umfangreiche Onlinehilfe" o. Ä.

Features aggregieren gewöhnlich zusammengehörige Anforderungen zu einem sinnvollen Ganzen (z. B. eine Menge mehrerer Anwendungsfälle, die gemeinsam alle Lebenszyklen eines geschäftlichen Objektes abdecken).

Unscharfe Featuredefinitionen

Die Definitionen zu Features in der Literatur und im Internet sind so unscharf wie unterschiedlich. Was liegt also näher, als diese Unschärfe zu akzeptieren und eine abstrakte Begriffsbestimmung zu verwenden, wie wir es in [OestereichWeissSchröder2004 [1]] oder [OestereichWeiss2007 [2]] getan haben:

Ein Feature ist eine Sammlung von (häufig noch zu konkretisierenden) Anforderungen.

Zunächst kann also alles Mögliche ein Feature sein. Klar ist lediglich, dass es sich stets um eine Aggregation irgendwelcher Anforderungen handelt.

Diese allgemeine Definition hilft jedoch nicht weiter, weswegen sinnvolle Featuredefinitionen immer im Kontext eines spezielles Zweckes stehen müssen. Die oben unterschiedenen Arten von Features oreintieren sich daher immer an einen speziellen Zweck. Erst dadurch gewinnen sie konzeptionellen Nutzen für uns.

Definition (Überblick)

Eine Feature beschreibt eine Eigenschaft (i.d.R. eine Funktionalität) eines Produktes oder Systems,

- die für den Anwender oder Käufer des Systems einen kaufentscheidenden Wert oder geschäftlichen Nutzen darstellt, der
- priorisierbar ist,
- deren Herstellung planbar ist
- und die so abstrakt sein kann, dass sie noch nicht objektiv prüfbar und abnehmbar ist, also noch keine Anforderung im engeren Sinne darstellt.

Im Kontext von Entwicklungsvorhaben ist die Unterscheidung folgender Arten von Features sinnvoll:

- Produktfeature: eine priorisierbare und zur Produktkonzeption brauchbare Einheit

- Vertragsfeature: ein vertragstaugliches Feature
- Releasefeature: eine sinnvolle Auslieferungs- bzw. Nutzungseinheit
- Iterationsfeature: eine brauchbare Umsetzungseinheit

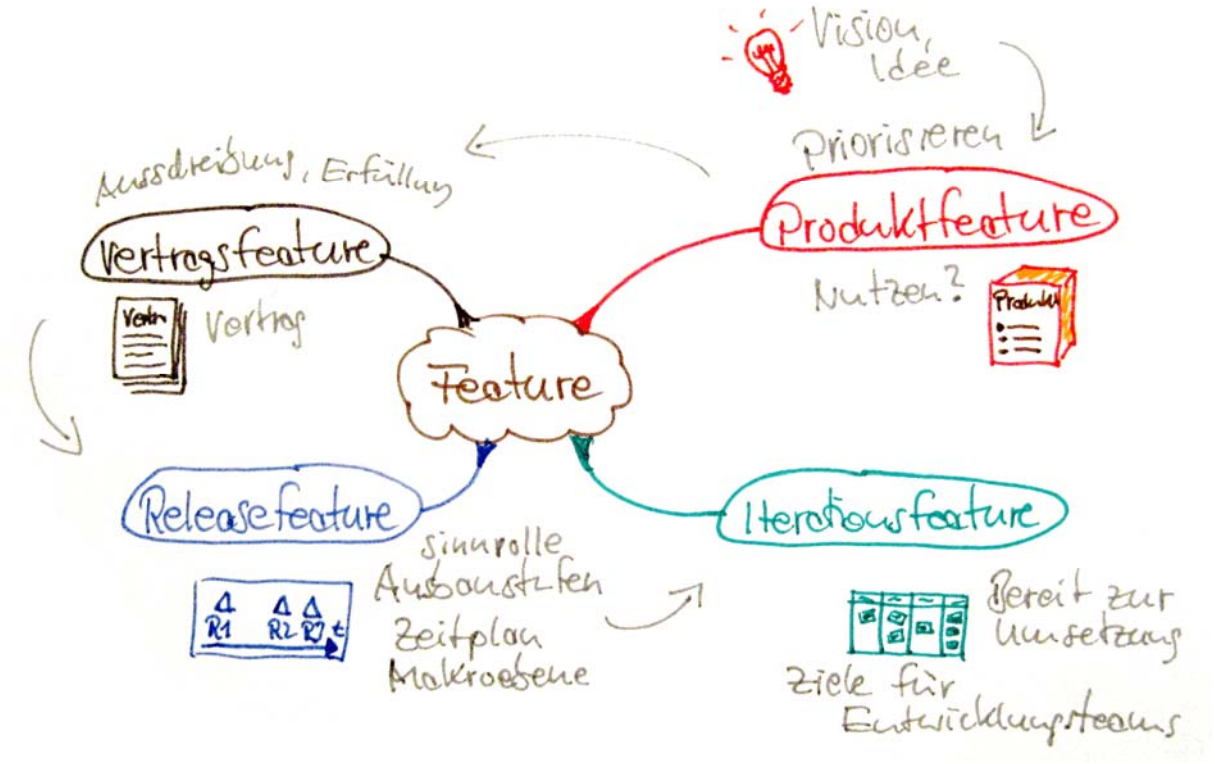

Abbildung 21: Die verschiedenen Arten von Features: Produktfeatures, Vertragsfeatures, Releasefeatures und Iterationsfeatures

Die verschiedenen Arten von Features beziehen sich jeweils auf einen speziellen Aspekt, einen Zweck und darum schließen sie sich nicht gegenseitig aus, sondern ein konkretes Feature kann gleichzeitig mehrere Aspekte erfüllen, bspw. sowohl ein Produktfeature als auch Vertragsfeature sein.

Produktfeature: abstraktes, priorisierbares und zur Produktkonzeption geeignetes Feature

Ein Produktfeature ist eine abstrakte, noch zu konkretisierende Anforderung, also noch keine Anforderung im engeren Sinne. Es dient zur Konkretisierung der Produktidee und Vision. Die Granularität von Produktfeatures entspricht ca. der auf typischen Produktkartons.

Der Zweck eines abstrakten Features ist die Beurteilung und Priorisierung des geschäftlichen Nutzens, um eine Menge wirklich relevanter Features zu einem wertvollen Produkt zusammenzustellen.

Vertragsfeature: vertragstaugliches Feature

Damit ein Feature Gegenstand von Verträgen, Ausschreibungen, Releases u. Ä. werden kann, muss es objektiv überprüfbar und abnehmbar sein. Deswegen unterscheiden wir vertragstaugliche von noch nicht vertragstauglichen Features.

Der Zweck eines vertragstauglichen Features ist die Berücksichtigung in einem Vertrag und die damit verbundene spätere objektive Feststellung der Vertragserfüllung.

Iterationsfeature: iterations- und teamspezifsiche Umsetzungseinheit

Ein Iterationsfeature stellt eine Menge inhaltlich zusammengehörender Anforderungen dar, die von einem Entwicklungsteam in maximal einer Iteration umsetzbar ist und die für den Anwender bzw. Kunden eine wahrnehmbare, sinnvolle und bzgl. des geschäftlichen Nutzens eigenständig bewertbare

sinnvolle Teilfunktionalität des zu entwickelnden Produktes darstellt. Es ist nicht notwendig, dass diese Teilfunktionalität auch unabhängig von anderen Features sinnvoll produktiv genutzt werden könnte (vgl. Releasefeature).

Die Menge aller Iterationsfeatures einer Iteration für ein Entwicklungsteam stellen die inhaltliche Zielvereinbarung mit dem Entwicklungsteam dar.

Der Zweck eines Iterationsfeatures ist die Kreation einer gut handhabbaren Planungseinheit oberhalb der Teamebene (d.h. oberhalb der internen Planung eines Entwicklungsteams) und die anschließende objektive Feststellung des Entwicklungsfortschrittes.

Die Größe eines Iterationsfeatures wird dadurch bestimmt, dass es von einem Entwicklungsteam in maximal einer Iteration umgesetzt werden kann und einen eigenständigen bewertbaren geschäftlichen Nutzen erzeugt. Die Größe ist also unter anderem abhängig von der Entwicklungsgeschwindigkeit (Velocity) bzw. von der Iterationsdauer, Teamgröße und Teamproduktivität (Velocity).

Benutzergeschichten, die bereit zur Umsetzung sind, sind Iterationsfeatures (siehe Benutzergeschichte und Inkrementbegutachtung).

Releasefeature: sinnvolle Auslieferungs- bzw. Nutzungseinheit

Releasefeatures stellen sinnvolle minimal nutzbare Ausbaustufen des Produktes dar. Sie aggregieren eine Menge inhaltlich zusammengehörender Detailanforderungen, die erstmalig Teil eines bestimmten Releases sind und die für die Benutzer bzw. den Kunden in ihrer Gesamtheit sinnvoll eingeführt und praktisch genutzt werden können.

Während bei einem Iterationsfeature der geschäftliche Nutzen lediglich abstrakt bewertbar und wahrnehmbar sein soll, soll ein Releasefeature auch praktisch sinnvoll nutzbar sein. So könnte ein Iterationsfeature das das Neuanlegen und Ändern von Kundendaten erlaubt, separat bzgl. des geschäftlichen Nutzens bewertbar und in einer Testumgebung sinnvoll testbar sein. Bevor diese Kundenverwaltung aber wirklich sinnvoll in der Produktionsumgebung und Praxis benutzt werden kann, ist möglicherweise ein anderes Iterationsfeature, bspw. die Übernahme der Altdaten aus dem Vorgängersystem notwendig. Beide zusammen bilden dann ein sinnvolles Releasefeature. Anderes Beispiel: Beim Bau eines Flugzeuges kann jede Tragfläche für sich ein sinnvolles Iterationsfeature darstellen, zum Fliegen werden aber beide benötigt.

Als Planungseinheit umfasst ein Releasefeature alle Iterationsfeatures, die bis zum Releasetermin erstellt werden sollen.

Der Zweck eines Releasefeatures ist die Zusammenstellung einer sinnvollen Auslieferungs- bzw. Nutzungseinheit.

Weiterführendes

Featurebezogene Akzeptanzkriterien

Damit ein Feature objektiv überprüfbar und abnehmbar wird, sind entsprechende Akzeptanzkriterien zu formulieren. Dabei sind die für ein Vertragsfeature relevanten Akzeptanzkriterien nicht unbedingt identisch mit den Akzeptanzkriterien eines Iterationsfeatures.

Sofern sie sich unterscheiden, sind die vertragsrelevanten meistens etwas abstrakter und Verständnistiefe und Detaillierung sind geringer als bei iterationsrelevanten Features.

Idealerweise und wahrscheinlich auch in einigen praktischen Fällen sind diese beiden Arten von Akzeptanzkriterien identisch - dennoch ist deren Unterscheidung hilfreich, um einfacher und früher Verträge schließen zu können. Wenn die iterationsrelevanten Features etwas konkreter sind als die vertragsrelevanten, dann wurde in die Klärung und Beschreibung iterationsrelevanten Features gewöhnlich auch mehr Aufwand und Zeit investiert und der entstandene Informationsbestand ist meistens größer. Sofern dieser Aufwand und dieser Informationsvorrat für die Gestaltung von Ausschreibungen, Projekt- und Werkverträgen u. Ä. nicht notwendig ist, sollte er vermieden werden. Dazu ist die Unterscheidung nützlich.

Der Zweck vertragsrelevanter Akzeptanzkriterien ist die Vereinbarung und Überprüfung eines Vertrages.

Der Zweck der iterationsrelevanten Abnahmekriterien ist die objektive Feststellung des Entwicklungsfortschrittes.

Der Zweck der releaserelevanten Abnahmekriterien ist die Feststellung, ob das Feature akzeptabel in das Produkt integriert ist.

Produktfeatures brauchen keine Akzeptanzkriterien, die Priorisierung und Nutzenbewertung für zur Entscheidung (Akzeptanz), ob das Produkt mit dem Feature ausgestattet werden soll oder nicht.

Abgrenzung

Diese Feature-Definition(en) stimmt nicht unbedingt mit der Definition von Feature in der Methode FDD überein.

Was Features nicht sind

Damit schließen wir folgende Anforderungen aus:

- Anforderungen, die sich nicht auf das Produkt, sondern darauf beziehen, wie das Produkt entsteht, also auf den Prozess (z. B. die "Software wird konform zum V-Modell XT entwickelt")
- Triviale Einzelfunktionen (z. B. "Dokument speichern") sowie triviale Einzelanforderungen (z. B. "das Angebotsdatum muss in der Zukunft liegen")
- Anforderungen, die auch nach weiterer Detaillierung (also prinzipiell) untestbar bleiben.
- Interne Anforderungen, die nicht kaufentscheidend sind, keinen direkten geschäftlichen Nutzen haben, sondern eine technische Bewandtnis haben (z. B. abgestimmte Datenmodelle)
- Einzelne Anwendungsfälle, die nur zusammen mit anderen einen Sinn ergeben (z. B. das Löschen eines Objektes)
- Rahmenbedingungen, die durch das Einhalten von Standards zu erfüllen sind (z. B. "die Anwendung soll in Java geschrieben sein")
- Nicht-Anforderungen (z. B. "kein dynamisches SQL verwenden")

Features fokussieren damit auf die wirklichen Leistungen des Produktes, d. h. überwiegend auf lauffähige Produktfunktionalität und damit auf das, was den eigentlichen Erfolg eines Projektes ausmachen sollte (und wofür der Auftraggeber auch gerne Geld ausgibt).

Anhang (benutzte Quellen etc.)

[1] [OestereichWeissSchröder2004] Buch
Bernd Oestereich, Christian Weiss, Claudia Schröder: *Objektorientierte Geschäftsprozessmodellierung mit der UML*
dpunkt.verlag, www.oogpm.de

[2] [OestereichWeiss2007] Buch
Bernd Oestereich, Christian Weiss, Claudia Schröder, Bernd Oestereich, Christian Weiss: *APM - Agiles Projektmanagement, Erfolgreiches Timeboxing für IT-Projekte*
dpunkt.verlag, http://www.dpunkt.de/buecher/2487/apm-%26%23150%3B-agiles-projektmanagement.html