Praxiswissen

Anforderungsmanagement

BUSINESS.
DATA.
TECHNOLOGY.

CINTELLIC
CONSULTING GROUP

Inhalt

www.cintellic.com
1. Auflage, 10/2018
von Paul Eisermann, Ksenia Buntova und Andrea Ferdinand

1. Definition einer Anforderung

Eine ***Anforderung*** ist eine Eigenschaft, die ein Gegenstand aus Sicht eines bestimmten ***Stakeholders*** aufweisen muss.
Gebe ich also ein IT-System in Auftrag, so beschreibt die Gesamtheit meiner Anforderungen, welche Eigenschaften das System aufweisen muss, damit der von mir benötigte Nutzen realisiert ist. Anforderungen werden stets aus Sicht eines bestimmten Stakeholders formuliert. Denn wo kein Nutznießer der Realisierung einer Anforderung vorhanden ist, ist sie auch nicht relevant.
Dabei unterscheidet man zwischen funktionalen und nicht-funktionalen Anforderungen. Eine **funktionale Anforderung** beschreibt, was ein System tun soll.

Beispiel: Das System muss beim Druck auf die Schaltfläche „alle Daten ausgeben" alle im Speicher vorhandenen Datensätze als Tabelle ausgeben.

Demgegenüber beschreiben ***nicht-funktionale Anforderungen***, wie ein System seine Funktionen erfüllen soll, ohne die eigentliche Funktionalität zu definieren. Gegenüber dem oben genannten Beispiel könnte man die folgende nicht-funktionale Anforderung erheben:

Beispiel: Das System soll bei der Ausgabe der Datensätze weniger als 0,001 Sekunden pro Datensatz benötigen.

In diesem Beispiel beschreibt die erste Anforderung eine Funktionalität, denn sie definiert ein Resultat der Arbeit des Systems als Folge eines bestimmten Inputs. Die zweite Anforderung tut genau dies nicht, legt dafür aber eine Eigenschaft dieser Funktionalität fest. Neben der Effizienz eines Systems zählen zu den nicht-funktionalen Anforderungen aber z. B. auch die Themen Änderbarkeit/Übertragbarkeit, Usability, Zuverlässigkeit sowie zu berücksichtigende ***Randbedingungen***.

2. Anforderungserhebung und -management als Prozess

Der Lebenszyklus einer Anforderung erstreckt sich über die gesamte Projektlaufzeit. Er beginnt in vielen Fällen mit der Definition eines ***Demands***. Dieser beschreibt kurz und knapp, welchen Nutzen ein Projekt realisieren soll und auf welchem Wege, mit welchen wesentlichen Randbedingungen und welchen Stakeholdern dieser Nutzen realisiert werden soll.
Zentrale Fragen sind in diesem Stadium:

- Was ist der Bedarf, der für das Vorhaben ausschlaggebend ist? (Welche Funktion fehlt im existierenden Prozess oder System? Was funktioniert im Arbeitsablauf nicht befriedigend?)

- Was ist der geschäftliche Nutzen des Vorhabens? (Welche relevante Größe soll verändert werden? Z. B.: Wie viel Kostenersparnis? Wie viel weniger manueller Aufwand? Wie viel zusätzlicher Umsatz?)
- Welche Anforderungen müssen zur Realisierung des Bedarfs und des Nutzens mindestens umgesetzt werden? (Grobanforderungen, die die beiden vorherigen Fragen auf die Perspektive des Systems beziehen)
- Wer ist betroffen? (Welche Einheiten im Unternehmen sind Stakeholder? Wer sind die relevanten Nutzer? Evtl.: Welche Umsysteme sind einzubinden?)

Darauf aufbauend tritt immer öfter die Definition eines ***Minimum Viable Product*** (MVP). Dieses beschreibt aus technischer Sicht, welchen Umfang ein durchzuführendes Projekt mindestens haben muss, um den zentralen Nutzen des Demands zu realisieren.

Diese beiden Arbeitsergebnisse sind im Anschluss Grundlage für die konzeptionelle Arbeit im Projektverlauf: Aus dem Demand werden von oder mit den Anforderern detaillierte Anforderungen abgeleitet. Das MVP kann je nach technischem Detailgrad zur Definition eines ***Solution Designs*** dienen.
Die mit dem Thema Anforderungen verbundenen Aufgaben sollten zu Beginn eines Projektes als Prozesse definiert und im Projektverlauf kontinuierlich weiterentwickelt werden. Diese Prozesse müssen primär folgende Aktivitäten abbilden:

- Dokumentation der fachlichen Anforderungen in einem Detailgrad, der für eine fehlerfreie Umsetzung ausreichend ist (vgl. *„5. Anforderungsschablonen“*)
- Abnahme der Anforderungen durch ein Entscheidungsgremium
- Sicherstellung der Testabdeckung
- Umgang mit Changes (Log, Genehmigung, Aktualisierung der Anforderungsdokumentation, Umsetzung, Testabdeckung)

Dabei ist grundsätzlich zwischen einem linearen Stufenmodell und einem iterativen Vorgehen zu unterscheiden (vgl. *„3. Wasserfall- und agiles Vorgehen im Vergleich“*).

3. Wasserfall- und agiles Vorgehen im Vergleich

Der Aufbau des Anforderungsprozesses ist grundsätzlich davon abhängig, ob in einem Projekt im ***Wasserfallmodell*** („klassisch“) oder agil, d. h. nach ***Scrum*** oder einer verwandten Methode vorgegangen wird.

Im Wasserfallmodell werden Anforderungen in einer frühen, zeitlich begrenzten Phase im Projektverlauf mit den Stakeholdern definiert und im Detail dokumentiert und in der Regel nur noch in Ausnahmefällen zu einem späteren Zeitpunkt verändert.
Agile Projekte beginnen mit einer groben Erfassung auf Basis eines Demands und analysieren diese groben Anforderungen, d. h. sie zerlegen sie in separate Pakete, die dann unabhängig voneinander umgesetzt werden können. Die Feinkonzeption geschieht dann iterativ im Projektverlauf abhängig von der Sprintplanung.

Eine saubere Arbeit mit Anforderungen, die Probleme im weiteren Projektverlauf minimiert, muss daher je nach Projektmethode unterschiedliche Prioritäten setzen:

In der Analyse- und Konzeptionsphase eines Wasserfallmodell-Projektes ist neben der vollständigen und korrekten Erfassung der Anforderung primär zu gewährleisten, dass ein möglichst detailliertes und widerspruchsfreies Konzept für das zu erstellende Produkt vorhanden ist und dass offene Punkte so weit wie möglich beseitigt sind. Im weiteren Projektverlauf sind Änderungen auf ihren Einfluss auf das Gesamtkonzept hin zu analysieren, in einem Changelog zu erfassen und als nachvollziehbare Änderungen in die Anforderungsdokumente einzupflegen.

In einem ***agilen Projekt*** ist zu Beginn zu gewährleisten, dass die grobe initiale Anforderungsdokumentation die tatsächlichen Anforderungen vollständig abdeckt. D. h. es ist eine sorgfältige Abgrenzung zum Out-of-Scope-Bereich vorzunehmen und die Anforderungen sind ihrem Sinn nach transparent zu machen, sodass für alle Beteiligten eine korrekte Interpretation möglich ist. Gemeinsam mit den Realisierern ist eine Methode zu vereinbaren, nach denen die Anforderungen in separat umsetzbare Teile (i. d. R. User Stories) gegliedert werden kann. Diese Teile müssen dann in einem ***Backlog*** erfasst und priorisiert werden.

Im weiteren Projektverlauf ist dieses Backlog zu pflegen. Die Sprints müssen so geplant werden, dass zum Beginn einer Umsetzung eine hinreichend verfeinerte Konzeption der umzusetzenden Bestandteile bereitsteht, d. h. die Anforderungsanalyse muss der Umsetzung immer um mindestens einen Sprint voraus sein.
Da bei einem agilen Vorgehen die enge Zusammenarbeit von Anforderern und Realisierern im gesamten Projektverlauf im Mittelpunkt steht, müssen die mit der Anforderungsanalyse betrauten Personen das Projektteam auch während eines laufenden Sprints bei dem Abgleich von Anforderung und Realisierung begleiten und im Nachgang mögliche Korrekturen definieren und treiben.

4. Qualitätskriterien für Anforderungen

Die Dokumentation von Anforderungen dient im Entwicklungsprozess einer Reihe von Zielen:

- Sie hilft dem Anforderer, eine klare Produktvision zu generieren und offene Punkte aufzudecken.
- Sie hilft dem Anforderer, Änderungen oder Ergänzungen der Anforderungen logistisch aufzuarbeiten.
- Sie hilft dem Anforderer bei der Kommunikation des Auftrags gegenüber dem Realisierer.
- Sie hilft dem Realisierer, die Bedürfnisse des Anforderers und dessen Auftrag genau zu verstehen.

- Sie hilft Anforderer und Realisierer, die Erfüllung der Anforderungen systematisch zu überprüfen.

Die wichtigsten Qualitätskriterien der Anforderungsdokumentation, um diese Ziele zu erfüllen, sind:

- Korrektheit: Die Anforderung entspricht inhaltlich dem, was der Anforderer meint.
- Eindeutigkeit: Eine Anforderung muss so definiert sein, dass kein Interpretationsspielraum bleibt.
- Vollständigkeit: Eine Anforderung muss so definiert sein, dass alle impliziten Bestandteile der Anforderung unerwähnt oder unklar bleiben.
- Testbarkeit: Aus einer Anforderung müssen Testfälle ableitbar sein, mithilfe derer die Erfüllung der Anforderung (wiederum eindeutig und vollständig) überprüfbar ist.
- Abstrahierbarkeit: Die Anforderung ist unabhängig von einer konkreten Implementierung formuliert.
- Widerspruchsfreiheit: Die Anforderung widerspricht inhaltlich keiner anderen Anforderung.

Zusätzlich sollten eine Reihe sekundärer Qualitätskriterien beachtet werden. Diese sind nicht kritisch für eine korrekte Umsetzung, erleichtern aber ein reibungsloses Anforderungsmanagement:

- Redundanzfreiheit: Die einzelnen Anforderungen überschneiden sich inhaltlich nicht.
- Angemessenheit: Die Anforderung ist unter Berücksichtigung der Rahmenbedingungen sinnvoll.

5. Anforderungsschablonen

Was ist eine Satzschablone?

Eine ***Satzschablone*** ist ein Satzbauschema für eine einzelne Anforderung. Sie gibt die syntaktische Struktur einer Anforderung vor und ermöglicht es dadurch, ein höheres Maß an Eindeutigkeit herzustellen, als das oft in eigenen Formulierungen der Fall ist.

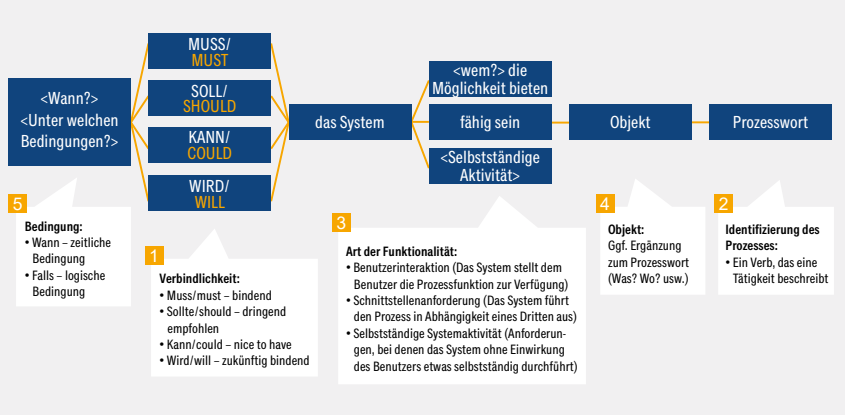
Warum Satzschablonen?

Eine in natürlicher Sprache formulierte Anforderung ist universal und kann von allen Stakeholdern ohne besondere methodische Kenntnisse verstanden werden. Dabei ist aber zu beachten, dass die natürliche Sprache mehrdeutig sein kann, was sowohl durch Unterschiede in Wissen, Kompetenzen und sozialer Prägung der Menschen als auch durch individuelle Wahrnehmungen verursacht wird. Um diesen negativen Effekt bei der Anforderungsdokumentation zu vermeiden, ist es zu empfehlen, Satzschablonen zu verwenden.

Weitere Vorteile der Verwendung sind dabei erhaltene Einheitlichkeit und Vollständigkeit der Anforderungen. Unkompliziert und einheitlich geschriebene Anforderungen

- sind einfacher und schneller zu verstehen,
- legen den Fokus auf Funktionalitäten und nicht auf Formulierung und Ausdrucksweise,
- ermöglichen es, alle Aspekte einer Anforderung zu berücksichtigen,
- reduzieren den Aufwand einer potenziellen Informations- oder Fehlersuche.

Wie sieht eine Satzschablone aus?



Wie gehe ich vor?

1. Wichtigkeit/Verbindlichkeit festlegen.
2. Prozess identifizieren.
3. Art der Funktionalität festlegen.
4. Objekt identifizieren.
5. Bedingung definieren.
6. Gesamte Anforderungen überprüfen.

Was gibt es noch zu beachten?

Folgende Tipps sind hilfreich, um Mehrdeutigkeiten in der Formulierung von Anforderungen zu vermeiden:

Nur eine Anforderung pro Satz formulieren!

Achten Sie von Beginn an darauf, jede Anforderung in einem eigenen Satz explizit zu machen. Überprüfen Sie jeden Satz darauf, ob sich nicht weitere Anforderungen hinter einzelnen Wörtern oder Formulierungen verbergen.

Kurze Sätze verwenden!

Lange Satzkonstruktionen sind nicht nur eine Gefahr für die Eindeutigkeit und Verständlichkeit von Anforderungen. Sie sind auch oft der Tatsache geschuldet, dass die Anforderungen nicht sorgfältig analysiert und daher nicht sauber getrennt formuliert wurden.

Anforderungen im Aktiv formulieren!

Bei Formulierungen im Passiv ist in der Regel unklar, wer oder was eine Aktivität ausführt. Schreiben Sie konsequent im Aktiv, legen Sie damit die Akteure und Verantwortlichen fest.

Nominalisierungen auflösen!

Hinter einer Nominalisierung verstecken sich häufig ein oder mehrere Prozessschritte, die zusätzliche Funktionalitäten des Sys-

tems darstellen. Werden Sie nicht genau und eindeutig definiert, besteht die Gefahr, dass sie nicht umgesetzt werden.

Wörter und Formulierungen mit eindeutiger Bedeutung verwenden!

Bei der Anforderungsformulierung sind grundsätzlich Mehrdeutigkeiten zu vermeiden (vgl. *„5. Anforderungsschablonen“*). Beispiele für Wörter, die Anforderungen uneindeutig machen, sind Adjektive wie „häufig“ oder „gut“. Stattdessen sollte ein Kriterium angegeben werden, wie häufig oder gut die Aktivität ausgeführt werden muss, um der Anforderung zu genügen.
Auch vermeintlich eindeutige Substantive wie „Benutzer“ sind von dieser Problematik betroffen: Stellen Sie sicher, dass klar ist, wer im jeweiligen Fall als Benutzer gilt.

7. Anforderungs-modellierung

Was sind Anforderungsmodelle?

Allgemein ist ein Modell ein vereinfachtes Abbild einer existierenden oder zu schaffenden Realität. Ein ***Anforderungsmodell*** stellt ein gewünschtes System mit einer Menge eindeutig definierter Relations- und Eigenschaftstypen dar. Dazu werden Diagramme sowie Ergänzungen in Textform verwendet.

Warum ist Anforderungsmodellierung sinnvoll?

- In Diagrammen visualisierte Sachverhalte sind für den Menschen **besser verständlich** und einfacher zu merken als textuelle Inhalte.
- Durch den vordefinierten Verwendungszweck jeder Modellart wird eine **klare Fokussierung** geschaffen.
- Da die Modellierungssprachen stark formalisiert sind, wird im Vergleich zur natürlichen Sprache das **Risiko für Mehrdeutigkeiten minimiert**.
- Durch den Formalisierungsgrad besteht oft die **Möglichkeit zur maschinellen Weiterverarbeitung**.
- Die grafische Darstellung ermöglicht eine bessere Nachvollziehbarkeit und **Reduzierung der Komplexität** im Vergleich zu langen Texten.

Wie werden Anforderungen modelliert?

Zur Modellierung von Anforderungen werden Modellierungssprachen verwendet. Modellierungssprachen sind über deren Syntax und Semantik klar definiert.

Syntax	Semantik
• Die Elemente, die verwendet werden dürfen • Die erlaubten Kombinationen der Elemente	• Die Bedeutung der Elemente • Die Basis für die spätere Interpretation des Modells

Im System- und Entwicklungskontext handelt es sich in diesem Fall häufig um die ***Unified Modeling Language*** (kurz ***UML***). Mit UML lassen sich viele verschiedene Diagrammtypen für unterschiedliche Zwecke abbilden.

Zur Modellierung der Anforderungen kann man grob zwei Diagrammarten differenzieren, die unterschiedliche Schwerpunkte abbilden:

- Strukturdiagramme
- Verhaltensdiagramme

Es kann durchaus sinnvoll sein, ein oder mehrere Modelle für jede dieser Perspektiven zu erstellen, um Anforderungen an eine Entwicklung ganzheitlich zu spezifizieren. Darüber hinaus können die Modelle durch textuelle Beschreibungen ergänzt werden oder diese konkretisieren (siehe auch Satzschablone).
Einfache Modelle lassen sich z. B. mit Microsoft Visio gut abbilden. Darüber hinaus gibt es eine Reihe weiterer kostenfreier oder gebührenpflichtiger Tools, die die Modellierung von UML-Diagrammen unterstützen.

8. Die wichtigsten UML-Diagramme im Überblick

- UML-Klassendiagramm (Strukturdiagramm)
- UML-Use-Case-Diagramm (Verhaltensdiagramm)
- UML-Aktivitätsdiagramm (Verhaltensdiagramm)

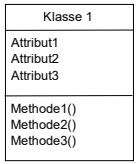
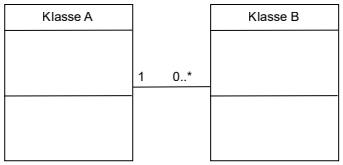
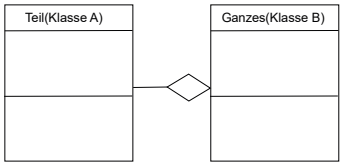
8 a. UML-Klassendiagramm

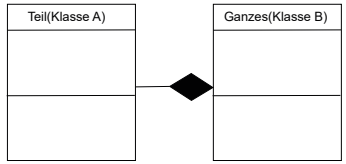
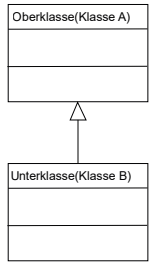
(Strukturdiagramm)

Beschreibung

Ein ***Klassendiagramm*** dient auf der Strukturebene zur grafischen Modellierung von Klassen, Schnittstellen und deren Beziehungen. Eine eKlasse bündelt Objekte mit gleicher Struktur und gleichem Verhalten. Z. B. könnte es eine Klasse „Kunde“ geben: jeder Kunde wird beschrieben durch z. B. Vorname, Nachname, E-Mail-Adresse und Geburtsdatum. Jeder Kunde muss darüber hinaus ein Kundenkonto (weitere Klasse) haben, welches wiederum z. B. durch eine Kontonummer und ein Eröffnungsdatum beschrieben wird.

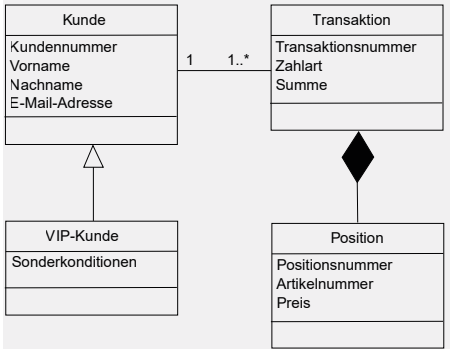
Wichtigste Elemente

Element	Beschreibung
Klasse 	Objekt mit gleichen beschreibenden Eigenschaften (Attributen) und bei Bedarf Methoden / Operationen
Assoziation mit Multiplizitäten 	Verbindung zur Beschreibung der Beziehung zwischen zwei Klassen Zahlen-Beschriftung definiert minimale und maximale Anzahl der Beziehungen, die ein Objekt bzgl. einer Assoziation eingehen kann, hier: • Klasse A kann 0 bis unendlich vielen Objekten von Klasse B zugeordnet sein • Klasse B ist immer genau einem Objekt von Klasse A zugeordnet
Aggregation (Teil-Ganzes Beziehung) 	Spezielle Form der Assoziation, A ist ein Teil von B

Element	Beschreibung
Komposition (Teil-Ganzes Beziehung) 	Spezielle Form der Assoziation, A ist ein Teil von B und existiert unabhängig von B nicht selbstständig (z. B. Räume sind Teil eines Gebäudes, aber ohne Gebäude gibt es auch die Räume nicht)
Generalisierung 	B ist eine spezielle Form von A (Unterklasse) und erbt alle Eigenschaften der Oberklasse, wird ggf. um eigene Attribute erweitert

Beispiel

Ein Kunde kann 1 bis unendlich viele Transaktionen durchführen. Eine Transaktion ist immer eindeutig einem Kunden zugeordnet.
Ein VIP-Kunde ist eine spezielle Art eines Kunden und wird zusätzlich zu den Standard Kunden-Attributen durch das Attribut Sonderkonditionen beschrieben.
Eine Position ist ein Teil einer Transaktion. Es gibt keine Position ohne Transaktion. Eine Transaktion besteht immer mindestens aus einer Position.



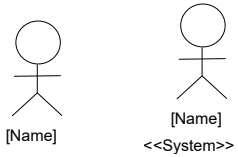
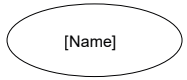
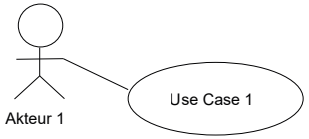
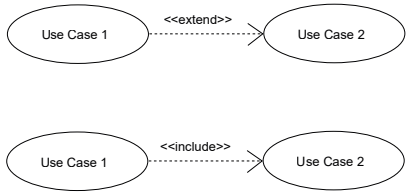
8 b. UML-Use-Case-Diagramm

(Verhaltensdiagramm)

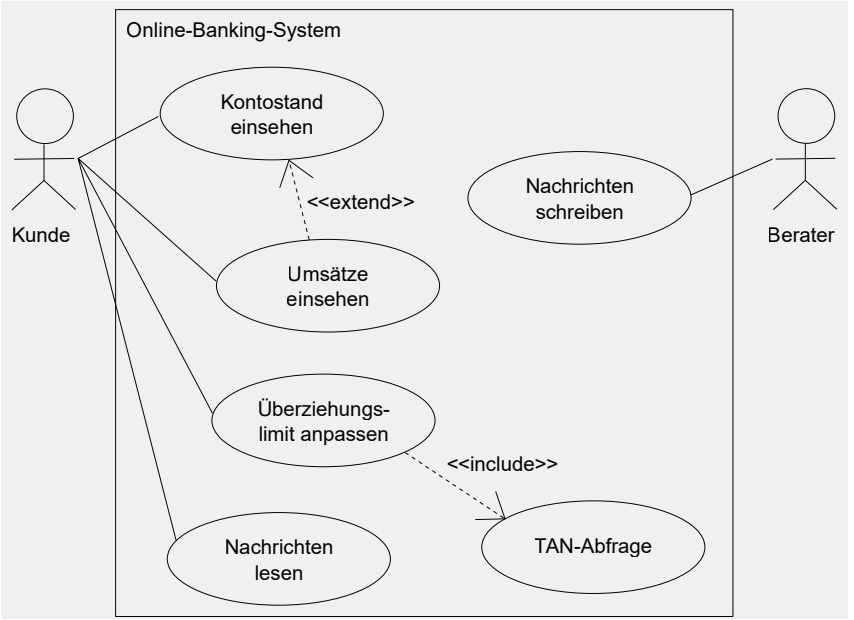
Beschreibung

Das UML-Use-Case-Diagramm dient Beschreibung des Systemverhaltens durch Anwendungsfälle, die über Verbindungen mit der Systemumwelt interagieren. In diesem Diagrammtyp wird das nach außen sichtbare Verhalten eines Systems sowie die Abgrenzung des Systems zur Umwelt dargestellt.

Wichtigste Elemente

Element	Beschreibung
System und Systemgrenze	Beschreibt das zu entwickelnde bzw. anzupassende System
Akteure Personen-Akteur / System-Akteur 	Stehen außerhalb des Systems und interagieren mit diesem
Anwendungsfall / Use Case 	Für das System definierte Anwendungsfälle
Assoziationen zwischen Akteur und Use Case 	Assoziationen sind keine Datenflüsse, sie beschreiben stattdessen Kommunikations-Beziehungen zwischen den Akteuren und dem System
Beziehungen zwischen Use Cases 	Extend = erweitert: Use Case 1 erweitert bei Eintreten einer Bedingung Use Case 2 (nicht immer zwingend erforderlich!) Include = inkludiert / beinhaltet: Use Case 1 beinhaltet in jedem Fall auch die Ausführung von Use Case 2

Beispiel



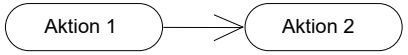
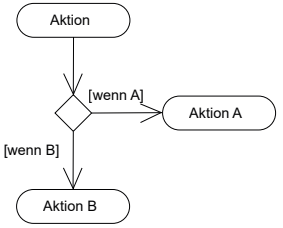
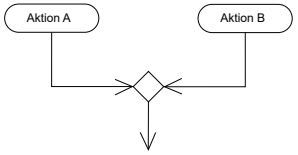
8 c. UML-Aktivitätsdiagramm

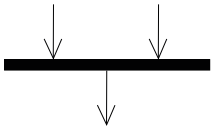
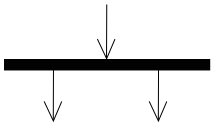
(Verhaltensdiagramm)

Beschreibung

Mithilfe von UML-***Aktivitätsdiagrammen*** werden dynamische Systemfunktionalitäten durch aufeinanderfolgende Prozesse und Ereignisse beschrieben. Eine Aktivität stellt einen einzelnen Schritt in einem Systemablauf dar und das Aktivitätsdiagramm bildet die Übergänge und Abhängigkeiten zwischen den einzelnen Aktivitäten ab.

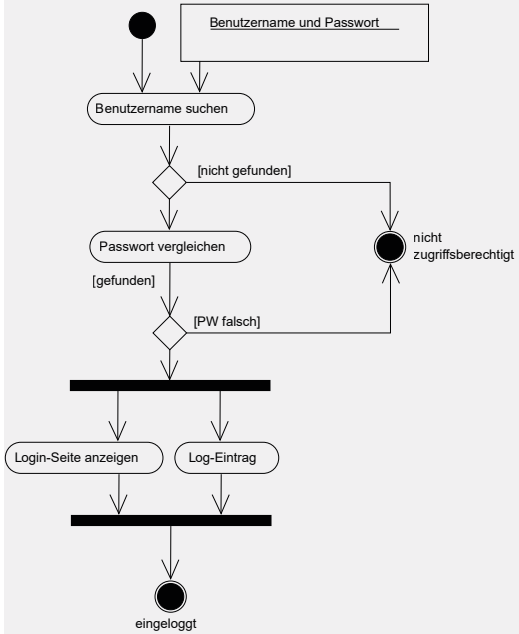
Wichtigste Elemente

Element	Beschreibung
Startknoten  Endknoten 	Der Startknoten repräsentiert das Ereignis, das die Ausführung des Aktivitätsdiagramms initiiert. Endknoten repräsentiert die Terminierung der Aktivität. Eine Aktivität muss mindestens einen Start- und einen Endknoten enthalten.
	Eine Aktion des Aktivitätsdiagramms
	Kontrollflüsse verbinden die Aktivitätsknoten (Start-/Endknoten sowie Aktionen), repräsentieren die logische Abfolge
Entscheidung mit Bedingungen  Zusammenführung von Entscheidungen 	Zur Darstellung von alternativen Kontrollflüssen. Entscheidung, mit welcher Aktion es weiter geht, anhand der dargestellten Kriterien in den eckigen Klammern (Bedingungen)

Element	Beschreibung
Nebenläufigkeit / Parallelisierung  Zusammenführung der Nebenläufigkeit 	Zur Darstellung von parallel angestoßenen und laufenden Aktionen

Beispiel

Login und Zugriffsberechtigung überprüfen



9. Weiterführende Links

Broschüre „Anforderungsmanagement“ von CINTELLIC als PDF-Download
https://www.cintellic.com/publikationen/anforderungsmanagement

Deutsche Website des International Requirements Engineering Board (IREB)
https://www.ireb.org/de

Übersicht über UML-Modellierungstools in der englischen Wikipedia
https://en.wikipedia.org/wiki/List_of_Unified_Modeling_Language_tools

10. Über CINTELLIC

Die CINTELLIC Consulting Group ist eine der führenden Unternehmensberatungen für die digitale Transformation von Kundenmanagement und CRM.
Das Beratungshaus fokussiert sich dabei auf die Schnittstelle der Themenfelder BUSINESS, DATA und TECHNOLOGY.

Ihre Mitarbeiter verfügen über langjährige Erfahrungen in den Bereichen Customer Relationship Management, Customer Experience Management, Marketing Operations Management, Kampagnenmanagement, Data Science und Business Intelligence.

Die Beratungsleistungen decken den gesamten Projektzyklus von der Konzeption, über die Realisierung bis hin zum Betrieb der Lösungen ab.

Darüber hinaus unterstützt CINTELLIC bei Entwicklung übergeordneter Strategien.
CINTELLIC-Berater/innen sind in den zentralen Projektmethoden ausgebildet und u. a. nach IREB im ***Requirements Engineering*** sowie Projektmanagementmethoden wie z. B. Scrum zertifiziert. In den Projekten stellt CINTELLIC die passgenaue Definition, das effiziente Management und die korrekte Umsetzung von Anforderungen sicher. Als Schnittstelle zwischen Fachbereich und IT gewährleistet CINTELLIC darüber hinaus das bereichsübergreifende Anforderungsmanagement sowie eine bessere Kommunikation zwischen den Bereichen im Projekt.

11. Glossar

Agile Entwicklung: Softwareentwicklungsprozess, der lange Planungs- und Realisierungszyklen durch kurze Iterationen ersetzt, in denen IT- und Fachressourcen eng zusammenarbeiten

Aktivitätsdiagramm: UML-Diagrammtyp zur Beschreibung dynamischer Systemfunktionalitäten durch aufeinanderfolgende Prozessen und Ereignissen

Anforderung: Eigenschaft, die ein System besitzen soll

Anforderungsliste: Zentrales Dokument im Projekt, in dem Anforderungen einzeln erfasst und verwaltet werden

Anforderungsmodell: Eine besondere Art eines Modells, das mit Hilfe von Diagrammen und textuellen Ergänzungen zur Spezifikation von Anforderungen dient

Anwendungsfalldiagramm: siehe ***Use-Case-Diagramm***

Backlog: Kurz für ***Product Backlog***; in agilen Projekten die priorisierte Gesamtheit der Anforderungen (i. d. R. in Form von ***User Stories***), aus denen das ***Sprint Backlog*** befüllt wird

Businessanalyse: Die Erforschung und Dokumentation fachlicher Abläufe und Anforderungen mit dem Ziel, diese für andere Bereiche wie die Anwendungsentwicklung zugänglich zu machen

CINTELLIC: Ihr kompetenter Partner für ein erfolgreiches ***Projekt***

Demand: Vorstufe einer Anforderungsdokumentation, in der grob Inhalt, Ziele und Nutzen eines Vorhabens beschrieben sind

Epic: Grobe Beschreibung der Anforderungen für ein Entwicklungsprojekt; große übergeordnete User Story; klassischerweise in Form einer „Erzählung“, die das Vorhaben in einen konkreten geschäftlichen Kontext einbettet

Fachkonzept: Dokumententyp aus der klassischen Entwicklung; beschreibt ausführlich und detailliert in einer Kapitelstruktur die Anforderungen zu einem einzelnen Themenbereich

Feature: 1. Allgemein Eigenschaft einer Software, insbesondere in Unterscheidung zu anderen Lösungen; 2. insbesondere in ***Scrum*** eine Anforderung, die umfangreicher als eine einzelne User Story aber weniger umfangreich als ein ***Epic*** ist

Funktionale Anforderung: Jede Anforderung, die sich auf eine Funktionalität eines Systems bezieht; Angabe darüber, was das System können muss

Kanban: Agile Entwicklungsmethode, die den Schwerpunkt auf die Parallelisierung der Abarbeitung von Aufgaben legt, sodass hoch priorisierte Aufgaben schnell begonnen und Ressourcen effizienter genutzt werden können

Klassendiagramm: UML-Diagrammtyp zur Modellierung der Strukturebene von Anforderungen an ein System; Gegenstand sind Klassen, Schnittstellen und deren Beziehungen

Lastenheft: Bei klassischem Projektvorgehen die Dokumentation der von einer Software zu erfüllenden Anforderungen; Basis für die Erstellung von Angeboten und Budgetschätzungen

Minimum Viable Product: Kurz MVP, Softwarelösung, die die Anforderungen mit dem kleinstmöglichen Aufwand (d. h. ohne „Nice-to-haves“) umsetzt; Ursprünglich definiert als die Version einer Software, die mit dem geringsten Aufwand lauffähig gemacht werden kann, sodass auf dieser Basis eine Implementierung einzelner Anforderungen möglich ist

Modell: Abstrahiertes Abbild einer existierenden oder zu schaffenden Realität

Nicht-funktionale Anforderung: Jede Anforderung, die sich nicht bzw. nur indirekt auf eine Funktionalität des Systems bezieht; ist in der Regel nicht fachlicher Natur; wie ein System die Funktion erfüllen soll

Pflichtenheft: Bei klassischem Projektvorgehen die Dokumentation des Umsetzungskonzepts bzw. der Lösung mit Bezug auf ein ***Lastenheft***; beschreibt den vorgesehenen Umsetzungsweg sowie die grobe Ressourcen- und Zeitplanung

Product Backlog: In agilen Projekten die priorisierte Gesamtheit der Anforderungen (i. d. R. in Form von ***User Stories***), aus denen das ***Sprint Backlog*** befüllt wird

Randbedingung: Bedingung, unter denen ein Projekt stattfindet und die bei der Umsetzung berücksichtigt werden müssen; zählt zu den ***nicht-funktionalen Anforderungen***

Requirements Engineering: Das Überführen von Anforderungen in abstrakte Strukturen (i. d. R. ***Anforderungsmodelle***), die in ihrer Komplexität und Eindeutigkeit als Basis für eine Umsetzung in Software dienen können

Review: Veranstaltung, die in der Regel am Ende eines ***Sprints*** steht und bei der die Entwicklungsergebnisse vorgestellt und gemeinsam auf ihre Übereinstimmung mit den Anforderungen überprüft werden

Satzschablone: Satzbauschema für eine einzelne Anforderung für ein höheres Maß an Eindeutigkeit und Vollständigkeit

Scope: Umfang eines Projekts in Abgrenzung zu den nicht vom Projekt zu erfüllenden Aufgaben („out of scope“)

Scrum: Agile Entwicklungsmethode, die sich durch präzise Regeln und Rollenbeschreibungen auszeichnet

Solution Design: IT-Dokument, in dem spezifiziert wird, wie die Fachanforderungen umgesetzt werden sollen

Sprint Backlog: enthält die Aufgaben, die in dem jeweiligen ***Sprint*** bearbeitet werden sollen

Sprint: Entwicklungszyklus bei agilen Projekten; dauert in der Regel zwei bis drei Wochen

Stakeholder: Alle Personengruppen in der Organisation, die vom Projekt betroffen sind und daher ein Mitspracherecht haben

UML: ***Unified Modelling Language***

Unified Modelling Language: Kurz UML; genormte grafische Modellierungssprache zur Spezifikation, Konstruktion und Dokumentation von Software bzw. Systemen

Use Case: Dt. Anwendungsfall; modellhafte Beschreibung einer zweckhaften Interaktion des Benutzers mit dem System

Use-Case-Diagramm: Auch Anwendungsfalldiagramm, UML-Diagrammtyp zur Beschreibung des Systemverhaltens durch Anwendungsfälle, die über Verbindungen mit der Systemumwelt interagieren

User Story: Einzelne Anforderung, die konkret aus Sicht eines Stakeholders formuliert ist und fein genug von anderen Anforderungen isoliert wurde, sodass eine separate Implementierung (oft im Rahmen eines einzelnen Entwicklungszyklus) möglich ist

Vorstudie: Projektphase, die allen anderen Projektaktivitäten vorausgeht und in der die Ausgangsbedingungen eines Projekts geklärt werden; am Ende steht i. d. R. eine Handlungsempfehlung

Wasserfallmodell: „Klassisches“ Projektvorgehen, bei dem eine Konzeptionsphase einer Umsetzungsphase vorausgeht und beide Aktivitäten in der Regel getrennt voneinander stattfinden

Über CINTELLIC

Die 2010 gegründete CINTELLIC Consulting Group ist eine auf digitales Kundenmanagement spezialisierte Unternehmensberatung, die ihre Klienten vom ersten Konzept bis zur Umsetzung in der Praxis ganzheitlich begleitet. An den Standorten in Bonn, Frankfurt am Main und München arbeiten mehr als 50 Mitarbeiterinnen und Mitarbeiter.

Zu den Klienten zählen DAX-Konzerne, führende mittelständische Unternehmen und insbesondere zahlreiche sogenannte „Hidden Champions" mit den Branchenschwerpunkten Banken und Versicherungen, Telekommunikation, IT, Medien, Unterhaltung, Handel, E-Commerce, Versorger und Logistik.

www.cintellic.com

Ihre Ansprechpartner

Dr. Jörg Reinnarth
Geschäftsführer
joerg.reinnarth@cintellic.com

Stephan Klöckner
Senior Manager
stephan.kloeckner@cintellic.com

Cintellic GmbH
Remigiusstraße 16
53111 Bonn
t +49 228 92 18 20
info@cintellic.com
www.cintellic.com

CINTELLIC
CONSULTING GROUP