# Workflows des RUP & deren Artefakte

Der RUP definiert Workflows für die 9 Kernaufgaben (Disciplines) des Prozesses. Workflows erwarten gewisse Artefakte als Eingabe und produzieren als Ausgabe wiederum eine festgelegte Anzahl von Artefakten.

## Core Workflows

Decken die fachlichen und technischen Bereiche ab.

### Business Modelling

Ziel dieses Workflows ist es ein Verständnis des Geschäftsprozesses sowie für das Umfeld in dem das System eingesetzt werden soll zu bekommen. Außerdem sollen die derzeitig existierenden Probleme und deren Verbesserungspotential erkannt werden.

* Business Vision
* Business Use Case Model
  + Business Use Case Diagram
  + Business Use Cases (Template/Analysis & Design), Description (Basic Flow, Alternative Flow, Extension
  + Business Actors
  + Business Workers
* Domain Modell
  + Business Objects und ihre Beziehung untereinander

### Requirements

Das Hauptziel ist es sich mit den Kunden sowie den Stakeholdern darauf zu einigen, welche Funktionen das Zielsystem haben soll. Dies stellt eine erste Basis für die Planung der technischen Umsetzung sowie für eine grobe Schätzung der Entwicklungsdauer des Projektes dar und gibt den Entwicklern einen Überblick über die Anforderungen.

* Use Case Model
  + Use Case Diagram
  + Use Cases (Template /Analysis & Design), Description (Basic Flow, Alternative Flow, Extension)
  + Actors
  + Workers
* Vision
* Stakeholder Request
* Prototype

### Analysis and Design

Hierbei wird aus den Anforderungen das eigentliche System abgeleitet.

#### Analyse Model

Das Analyse Model ist Teil des RUP Workflows Analyse & Design. Aus den im Requirements Workflow erstellten Use Case Modellen wird das Analyse Model abgeleitet. Es beschreibt die fachliche Logik des zu entwickelnden Systems aus statischer sowie dynamischer Sicht. Dies erfolgt mittels UML Diagrammen. Hauptziel ist es eine stabile Grundstruktur für das System zu schaffen.

##### 3.1.3.1.1 Unterschiede zwischen Anforderungsanalyse (Requirements) und Analysemodel

Das Use Case Model unterscheidet sich vom Analysis Model durch einige wesentliche Punkte:

* Analysis Model Diagram (AMD) statische Sicht
  + Class Diagram, erweitert um die Analyseklassen
  + boundary classes (Schnittstellen)
  + control classes (Business Logic)
  + entity classes
* Use Case Realization
  + Precondition
  + Sequence Diagram Basic Flow, alternate flows
  + Participated classes
  + Participated entities

|  |  |
| --- | --- |
| **Use Case Model** | **Analyse Model** |
| bedient sich der Sprache des Kunden | der des Programmierers |
| beschreibt das System aus einer externen Sicht | zeigt die interne Darstellung (Logik) |
| erhält seine Struktur durch die Anwendungsfälle | hingegen durch die stereotypen Klassen und Module |
| dient als Vertrag zwischen Kunde und Entwickler | wird von den Entwicklern verwendet |
| kann Redundanzen und Inkonsistenzen enthalten | darf keine Redundanzen und Inkonsistenzen enthalten |
| stellt die Funktionalität dar | stellt deren Umsetzung dar |

### Implementation

In diesem Schritt findet die tatsächliche Implementierung des Systems statt. Bevor diese möglich ist, ist es notwendig den Implementierungsprozess zuerst ausreichend zu planen.

* Component Diagram
* Implementation Subsystem Diagram
* Integration Build Plan
* Build
* Implementation Model

### Test

Dient dem Aufspüren von möglichen Fehlern im System sowie der Validierung ob das System den Anforderungen des Kunden entspricht.

* Test Plan
* Test Script
* Test Log
* Test Class/Data
* Test Component

#### Arten von Tests

##### Unit-Test

Mittels Unit-Tests werden einzelne Komponenten der Software getestet. Dies erfolgt bereits im Implementation-Workflow.

##### Systemtest

Es werden alle neu hinzugekommenen oder veränderten Teile der Software geprüft. Systemtest benötigen immer mehrere Durchläufe. Daher bietet es sich an diese zu automatisieren. In größeren Firmen existieren eigene Abteilungen um Systemtests durchzuführen.

##### Integrationstest

Das gesamte System wird mit all seinen Komponenten geprüft.

##### User-Acceptance-Test

Der Endnutzer testet das fertige System. Dafür kommt meist das Blackbox-Verfahren zum Einsatz. Im Unterschied zum Whitebox-Verfahren wird hierbei nur auf das Verhalten der Software anstatt auf den eigentlichen Programmcode geschaut.

### Deployment

Das entwickelte System wird an den Kunden ausgeliefert.

* Deployment Plan
* Release Notes
* Product
* Training Materials
* End User Support Material

## Supporting Workflows

Beinhalten jene Tätigkeiten welche unabhängig von der Entwicklungsphase im gesamten Prozess durchzuführen sind.

### Configuration & Change Management

Auf Forderungen des Kunden nach Änderungen im bestehenden System muss entsprechend reagiert werden.

* Change Request
* Configuration Management Plan
* Configuration Audit Findings
* Workspace Integration und Development

### Project Management

Ein solides Projektmanagement soll aufgebaut werden um Prozesse des Projektes besser planen und ausführen zu können.

* Business Case
* Iteration Plan
* Risk List
* Quality Assurance Plan
* Work Order

### Environment

Es soll für das mit der Entwicklung der Software beauftragte Team optimale Bedingungen geschaffen werden. Dabei ist es vor allem wichtig die Entwickler mit den notwendigen Tools und Methoden auszustatten.

* Design Guidelines
* Test Guidelines
* User Interface Guidelines
* Tool Guidelines
* Development Case

## Vor- und Nachteile des RUP

**Vorteile:**

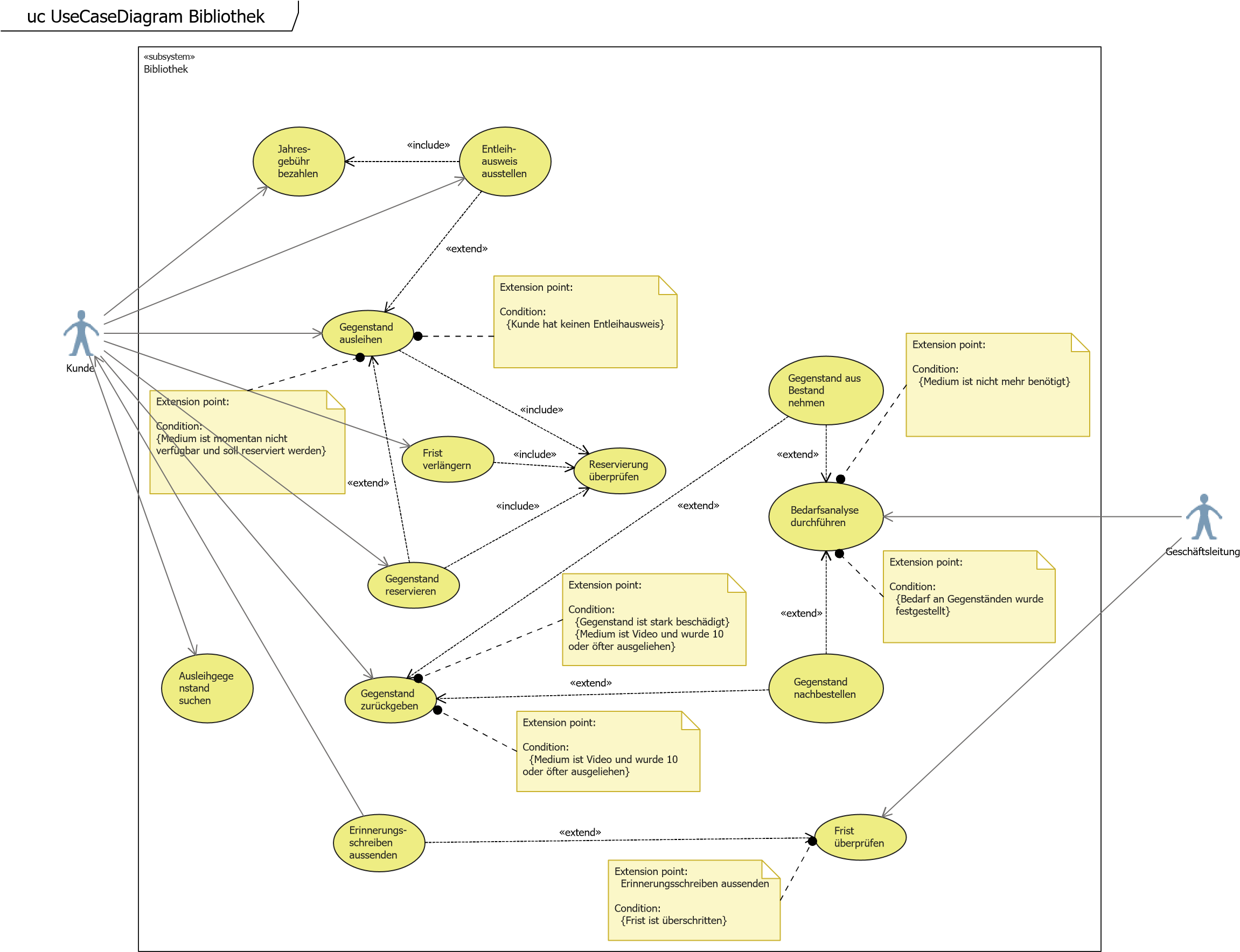
* Risikofaktoren können schnell erkannt werden.
* Bessere Arbeitsteilung anhand verschiedener Rollen
* Man kann sich auf mögliche Probleme konzentrieren
* Laufende Qualitätssicherung
* Prozess verbessert sich nach jeder Iteration

**Nachteile:**

* Komplexer Prozess
* Die Werkzeuge für RUP sind kostenpflichtig
* Oft komplizierte Iterationsplanung

# Beispiele

## Business Use Case Diagram



## Business Use Case Templates

**Use case:** Gegenstand ausleihen

**Ziel:** Kunde leiht sich einen Gegenstand aus

**Vorbedingung:** ISBN bekannt (UC: „Ausleihgegenstand suchen“ muss erfolgreich abgelaufen sein)

**Nachbedingung bei Erfolg:**

Kunde hat Gegenstand ausgeliehen

**Nachbedingung bei Fehlschlag:**

Kunde hat Gegenstand nicht ausgeliehen und Fehlermeldung an Kunden

**Akteure:** Kunde

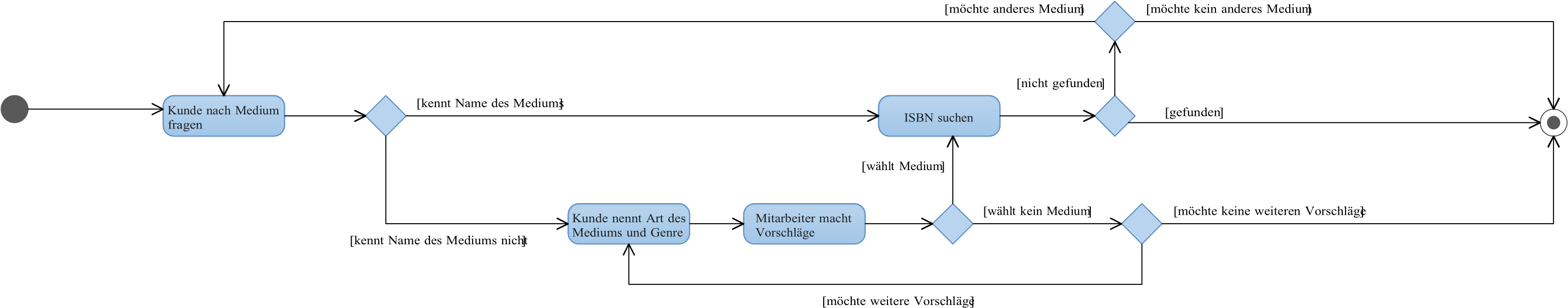
**Auslösendes Ereignis:**

Kunde will Gegenstand ausleihen

**Standard-Ablauf:**

1. Kunde wird nach Entleihausweis gefragt
2. Wenn Kunde keinen Entleihausweis hat, dann erhält er einen (Entleihausweis erhalten)
3. Kunde nennt Gegenstand, den er ausleihen möchte
4. Gegenstand existiert in Bibliothek
5. Reservierung wird überprüft (UC: Reservierung überprüfen)
6. Gegenstand ist frei
7. Kunde bezahlt Kaution
8. Daten des Kunden (Name, Anschrift) + Datum werden festgehalten
9. Kunde erhält Gegenstand
10. Kunde erhält Bestätigung

## Activity Diagram (Medium suchen)



# Quellenangabe

* <http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/is-management/Systementwicklung/Vorgehensmodell/Rational-Unified-Process-%28RUP%29/index.html>
* <https://de.wikipedia.org/wiki/Rational_Unified_Process>
* <https://en.wikipedia.org/wiki/Rational_Unified_Process>
* <https://airbrake.io/blog/sdlc/rational-unified-process>
* <http://members.it4education.at/fubbcontent/lektionen/letzte_lieferungen/Phasenmodelle/data/18.html>
* <https://www.ibm.com/developerworks/rational/library/content/03July/2000/2062/2062_Eeles3.pdf>
* <https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf>
* <https://books.google.at/books?id=ycn-AAAAQBAJ&pg=PA102&lpg=PA102&dq=disziplinen+des+rup&source=bl&ots=K0eej_TG0n&sig=KSNJwPVuuj-75fOnoyaLoSjJnB8&hl=de&sa=X&ved=0ahUKEwiir-mQ0_DWAhUESRoKHQCeDhQQ6AEIOTAD#v=onepage&q=disziplinen%20des%20rup&f=false>
* <https://www.slideshare.net/ERICEV/rup-2248862>