**Rational Unified Process (RUP)**

**5AHIFS 2017/18**

**Marcel Judth**

**Melanie Bugelnig**

Inhaltsverzeichnis

[Abstract 2](#_Toc535934111)

[1 Allgemein 4](#_Toc535934112)

[1.1 Entstehungsgeschichte 4](#_Toc535934113)

[1.2 Concept 4](#_Toc535934114)

[1.2.1 Role 6](#_Toc535934115)

[1.2.2 Artifact 6](#_Toc535934116)

[1.2.3 Activity 6](#_Toc535934117)

[2 Entwicklungsphasen und Iterationen 7](#_Toc535934118)

[2.1 Phasen 7](#_Toc535934119)

[2.1.1 Inception (Konzeptualisierungsphase): 7](#_Toc535934120)

[2.1.2 Elaboration (Entwurfs- und Ausarbeitungsphase): 7](#_Toc535934121)

[2.1.3 Construction (Implementierungsphase): 8](#_Toc535934122)

[2.1.4 Transition (Auslieferungsphase): 8](#_Toc535934123)

[2.2 Iterationen 9](#_Toc535934124)

[3 Workflows des RUP & deren Artefakte 10](#_Toc535934125)

[3.1 Core Workflows 10](#_Toc535934126)

[3.1.1 Business Modelling 10](#_Toc535934127)

[3.1.2 Requirements 10](#_Toc535934128)

[3.1.3 Analysis and Design 10](#_Toc535934129)

[3.1.4 Implementation 11](#_Toc535934130)

[3.1.5 Test 11](#_Toc535934131)

[3.1.6 Deployment 12](#_Toc535934132)

[3.2 Supporting Workflows 12](#_Toc535934133)

[3.2.1 Configuration & Change Management 12](#_Toc535934134)

[3.2.2 Project Management 12](#_Toc535934135)

[3.2.3 Environment 13](#_Toc535934136)

[3.3 Vor- und Nachteile des RUP 13](#_Toc535934137)

[4 Beispiele 14](#_Toc535934138)

[4.1 Business Use Case Diagram 14](#_Toc535934139)

[4.2 Business Use Case Templates 14](#_Toc535934140)

[4.3 Activity Diagram (Medium suchen) 15](#_Toc535934141)

[5 Quellenangabe 15](#_Toc535934142)

# Abstract

The Rational Unified Process (RUP) is a software development process, developed by IBM. It not only provides a software development process, but also the necessary tools and templates needed. The RUP is not designed to be used as is, but rather to be adapted as required for the circumstances it is applied in.

The first version of the Rational Unified Process was developed in 1996 by the RUP team of the company Rational Software under the name of Rational Objectory Process, but later renamed to Rational Unified Process in order to align the name to the Unified Modelling Language, which is an essential tool of the RUP. In 2003 IBM acquired the RUP from Rational Software.

The development process consists of four different phases, upon which the core workflows Business Modelling, Requirements, Analysis and Design, Implementation, Test and Deployment are distributed. The four phases are being iterated over several times until the result of the project meets all requirements. Each core workflow uses the documents of the previous workflow, also called artefacts as an input and produces its own artefacts as a result.

**Those four phases are:**

1. **Inception**

The main goal is to reach an agreement concerning the goal of the project between all stakeholders.

1. **Elaboration**

The basic architecture of the future system is being planned, which is an essential factor for the Construction phase to be successful.

1. **Construction**

The phase in which the actual programming takes place.

1. **Transition**

The finished product is handed over to the customer.

In addition to the core workflows there exist three so called supporting workflows. These workflows are vital to the whole project and not limited to the four phases.

The goals of the supporting workflows are being able to handle requirement changes, building a well organised project management as well as providing the development team with everything necessary for the successful realisation of the project.

**Advantages of RUP**

* Better risk management
* Efficient use of resources
* Issues are discovered early
* Enables the team to deliver exactly what the customer wants

**Disadvantages of RUP**

* Complex process
* An expert is necessary to fully implement the process
* Can easily get out of control
* The tools needed are not free

# Allgemein

Der Rational Unified Process (RUP) ist ein 1999 von Rational (heute IBM Rational) veröffentlichtes objektorientiertes, aktivitätsgetriebenes Vorgehensmodell. RUP ist stark von der Unified Modelling Language (UML) geprägt und bietet eine Methode zur Softwareentwicklung auf Basis der UML. Somit ist RUP ein spezifisches Vorgehensmodell für die UML-basierte Softwareentwicklung und wird vollständig von Tools unterstützt.

Der Rational Unified Process basiert auf folgenden 3 Grundprinzipien:

1. Anwendungsfälle (Use-Cases)
2. Architektur im Zentrum der Planung
3. inkrementelles und iteratives Vorgehen

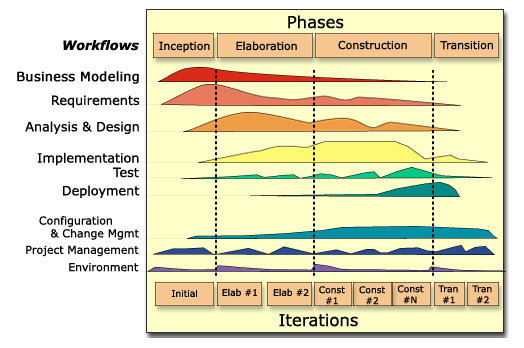
## Entstehungsgeschichte

Die Entstehung des RUP begann, als sich die drei Softwareentwickler des Unternehmens Rational Inc. Grady Booch, Ivar Jacobson und James Rumbaugh – auch als „The Three Amigos“ bekannt – Mitte der 1990er Jahre auf ein einheitliches Notationssystem einigten. Dieses Notationssystem ist heute als UML bekannt.

1996 entstand der Rational Objectory Process (ROP). Aufgrund der großen Abhängigkeit von der Unified Modelling Language wurde der ROP schließlich zu Unified Process umbenannt.

Eine konkrete Implementierung des Unified Process ist der Rational Unified Process. Die erste Version des RUP aus dem Jahre 1999 führte die Vorschläge dieser drei Begründer für eine einheitliche Modellierungsmethode zusammen.

## Concept



**Horizontal axis (development phases and iterations):**  
The RUP defines four Phases:

* Inception (Project setup, Concept)
* Elaboration (Draft, Architecture)
* Construction (Implementation)
* Transition (Deployment, Commissioning)

According to the iterative approach these phases are performed in iterations (Elab#1, Elab#2, Const#1, Const#2, …)

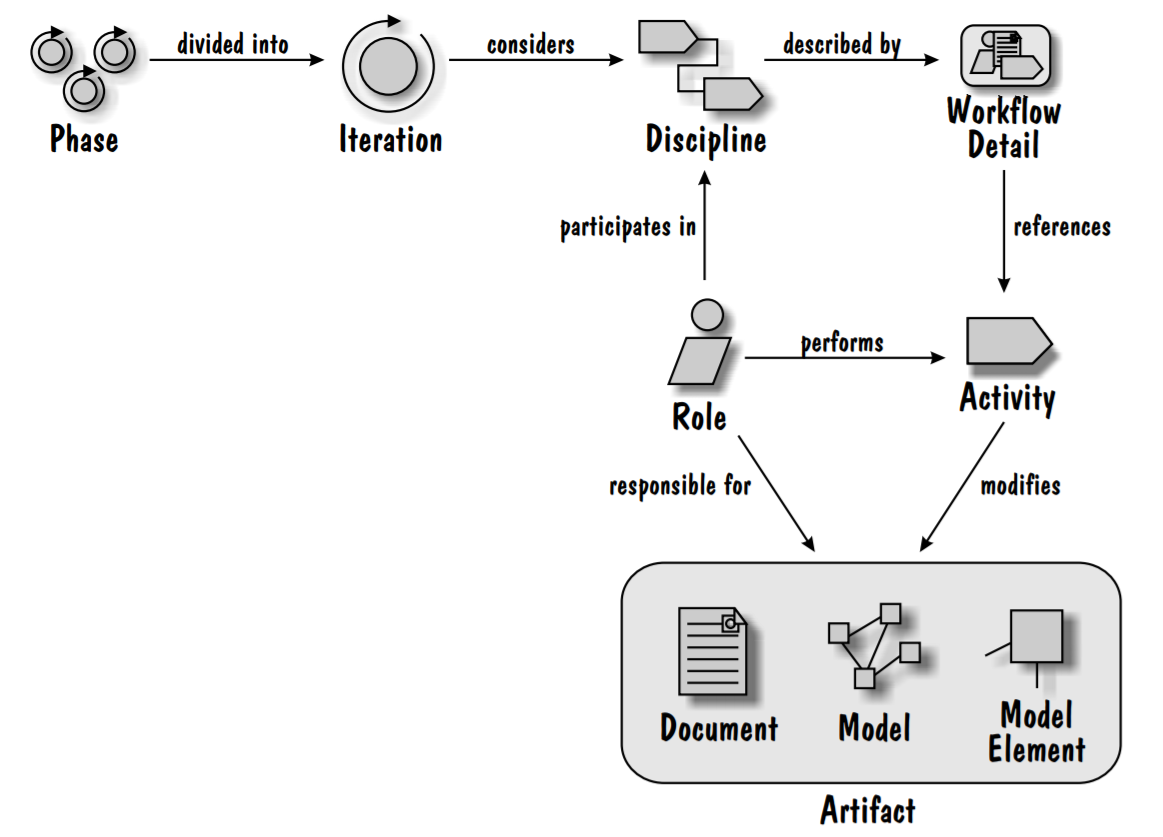
Each iteration results in a milestone. That is where it is decided if another iteration is required or if all set goals are achieved and the team can proceed to the next Iteration.  
  
**Vertical Axis (Disciplines / Workflows):**  
A project is divided into individual disciplines (activities). Each discipline is assigned a workflow, which is performed each iteration. A workflow represents a meaningful procedure for a successful execution of a discipline. Each step of a workflow should yield a part of an artefact.

**Colored Areas (Effort of disciplines per phase)**

The colored areas in the diagram represent the amount of work which goes into each discipline per phase.

An important fact is that disciplines are not restricted to a single phase. This means that unlike the “waterfall model” planning, development and testing does not take place strictly one after another. For example, using the RUP testing already happens during the Inception and Elaboration phases, and modelling / designing is still performed during the Construction phase.

Of course, certain disciplines / workflows are dominant during some phases but they are not performed exclusively in these phases. This is to ensure all requirements which are established near the end of the project can be considered. This difficulty has led to many projects (developed using the waterfall model) being abandoned.



### Role

= “Who”

A role defines the tasks (behavior) and responsibilities of a person or a team. A role performs a set of activities and is responsible for one or more artifacts.

The assignment of roles is done by the project manager.

### Artifact

= “What”

Artifacts are documents which are created and/or used by a role during the conduction of a process. They act as documentation or source of information. Some are meant for developers, others for customers.

Different artifacts will be listed and described in more detail with their corresponding workflows.

### Activity

= “How”

An activity is a unit of work which provides a meaningful result. IT has a clear purpose which usually involves creating or updating artifacts.   
Activities are assigned to certain roles.

Activities may be repeated several times, especially when executed in different iterations. They consist of one or multiple steps.

Example: Implementation of design elements:

1. Implement Design Subsystems
2. Implement Framework Components
3. Implement Design Classes and Interfaces
4. Implement Deployment Elements

# Entwicklungsphasen und Iterationen

Die Projektentwicklung gliedert sich gemäß dem RUP in vier Phasen, welche wiederrum aus mehreren Iterationen bestehen können.

## Phasen

### Inception (Konzeptualisierungsphase):

In dieser ersten Phase wird der Geschäftsfall etabliert und alle Stakeholder (Personen / Institutionen, die Vorteile aus der positiven Durchführung des Projektes ziehen) einigen sich auf die Ziele und den Umfang des Projekts.   
Der Hauptfokus in der Inception-Phase liegt darin festzustellen, ob sich die Umsetzung lohnt bzw. ob diese überhaupt möglich ist.

Ergebnis-Meilenstein: **Lifecycle Objectives Milestone**

*Aufgaben:*

* Projektvision formulieren
* Alle Use Cases identifizieren und die wichtigsten auszuarbeiten
* Festlegung von Projektumfang und der Projektdauer
* Definition von Abnahmekriterien (Pflichtenheft)
* Gesamtkosten und Zeitplan des Projekts abschätzen, Alternativen bewerten bezüglich
* Risikoanalyse
* Benötige Umgebung schaffen (Hardware, Software, Ressourcen, …)

*Ergebnisse bei erfolgreicher Durchführung:*

* Vision-Dokument
* Überblick über die wichtigsten Anforderungen
* Initiales Use Case Modell (10-20% aller Use Cases)
* Initiale Glossary (kann teilweise als Domain Model dargestellt sein)
* Erste Kostenschätzung

### Elaboration (Entwurfs- und Ausarbeitungsphase):

Ziel dieser Phase ist die Analyse von Anforderungen und der benötigten Systemarchitektur. Die Elaboration-Phase ist besonders kritisch, da mit ihrem Ende die bis hierhin wenig riskante Durchführung in der nächsten Phase, wo die Entwicklung / Programmierung stattfindet, stark risikobehaftet und kostenintensiv ist.

Ergebnis-Meilenstein: **Lifecycle Architecture Milestone**

*Aufgaben:*

* Vision auf Stabilität überprüfen
* Architektur auf Stabilität überprüfen
* Stimmen alle Stakeholder zu, dass die momentane Vision umgesetzt werden kann?
* Ist der Plan für die nächste Phase (Construction) ausreichend detailliert und genau?
* Ist der tatsächliche finanzielle Aufwand bezogen auf den geplanten zulässig?

*Ergebnisse:*

* Use Case Modell (ca. 80% vollständig) für alle Bereiche, speziell für die risikoreichen
* Verfeinerter Projektplan, der gegebene Rahmenbedingungen berücksichtigt (Hardware, Systemsoftware, Netzwerke, Datenbanken) und allfällig planbare Iterationen berücksichtigt
* Evtl. Ergebnisse von Tests mit Architekturprototypen
* Eine verfeinerte Risikoanalyse und Risikobewertung

### Construction (Implementierungsphase):

In dieser Phase werden alle übrigen Anforderungen geklärt und anschließend wird das gesamte System, basierend auf der Architekturplanung der Elaboration Phase, implementiert. In der Construction Phase gilt es eben dem Entwickeln neuer Softwarekomponenten auch bereits existierende Services oder Software zu integrieren.

Ergebnis-Meilenstein: **Initial Operational Capability Milestone**

*Aufgaben:*

* Implementierung der Software abschließen
* Software ausreichend testen
* Software auslieferungsfertig machen

*Ergebnisse:*

* Das System wurde entsprechend der Planung implementiert
* Alle benötigten Features sind vorhanden
* Akzeptanzkriterien wurden bei den Tests erfüllt

### Transition (Auslieferungsphase):

In der letzten Phase wird das entwickelte und getestete Produkt an die Kunden (Endnutzer) ausgeliefert. Hierbei liegt der Fokus auf der Installation, Konfiguration und Benutzung. Architektur- oder Strukturprobleme sollten nicht auftreten, da diese bereits in der Planungs- und Entwurfsphase geklärt wurden. Deshalb geht es hier nicht um größere Änderungen, sondern lediglich um das sogenannte „Fine-Tuning“. (Post-Release Support, Bug fixes, Patches)

Ergebnis-Meilenstein: **Product Release Milestone**

*Aufgabe:*

* System erfolgreich veröffentlichen

*Ergebnis:*

* Das System hat die Akzeptanzkriterien in der Umgebung des Endnutzers bestanden

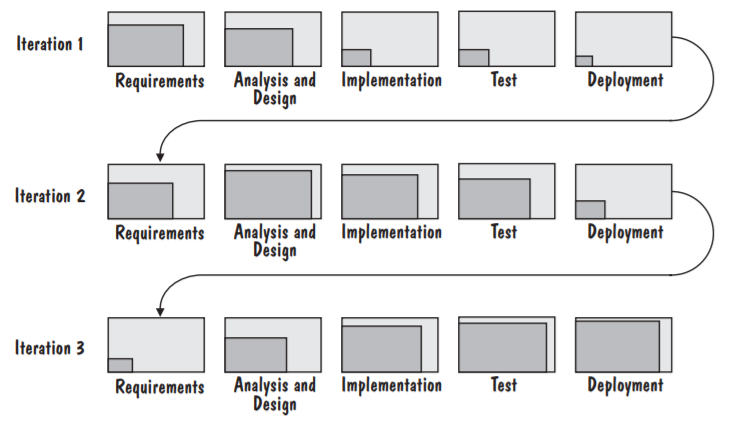
## Iterationen

Iterationen sind ein Schlüsselmerkmal von erfolgreicher Softwarenentwicklung. Deshalb werden die unter Punkt 2.1 beschriebenen Phasen üblicherweise in mehreren Iterationen durchlaufen.

Ziel einer Iteration ist es ein „Release“ zu erreichen. Dies kann ein interner Meilenstein sein aber auch ein neues Update für den Endnutzer. Um ein brauchbares Ergebnis zu erzielen ist es sinnvoll, pro Iteration alle Workflows zu Durchlaufen.

Entwickeln in Iterationen resultiert in „wachsender“ Software, d.h. das Produkt wird von Iteration zu Iteration ausgereiften. (besseres Verständnis der Anforderungen, robustere Architektur, erfahreneres Entwicklerteam, …)

Die folgende Grafik zeigt, wie im Laufe von Iterationen der Fokus auf verschiedene Workflows fällt. Es werden zwar alle Aktivitäten während jeder Iteration durchgeführt, aber die dafür aufgewandte Zeit verschiebt sich im Laufe der Iterationen. (Dargestellt durch die Grauen Boxen)



Jedem Release liegen diese Dokumente bei:

* Releasenotes (Beschreibung des Releases)
* Dokumentation für den Benutzer
* Changelog (Änderungen zum vorherigen Release)

# 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:**

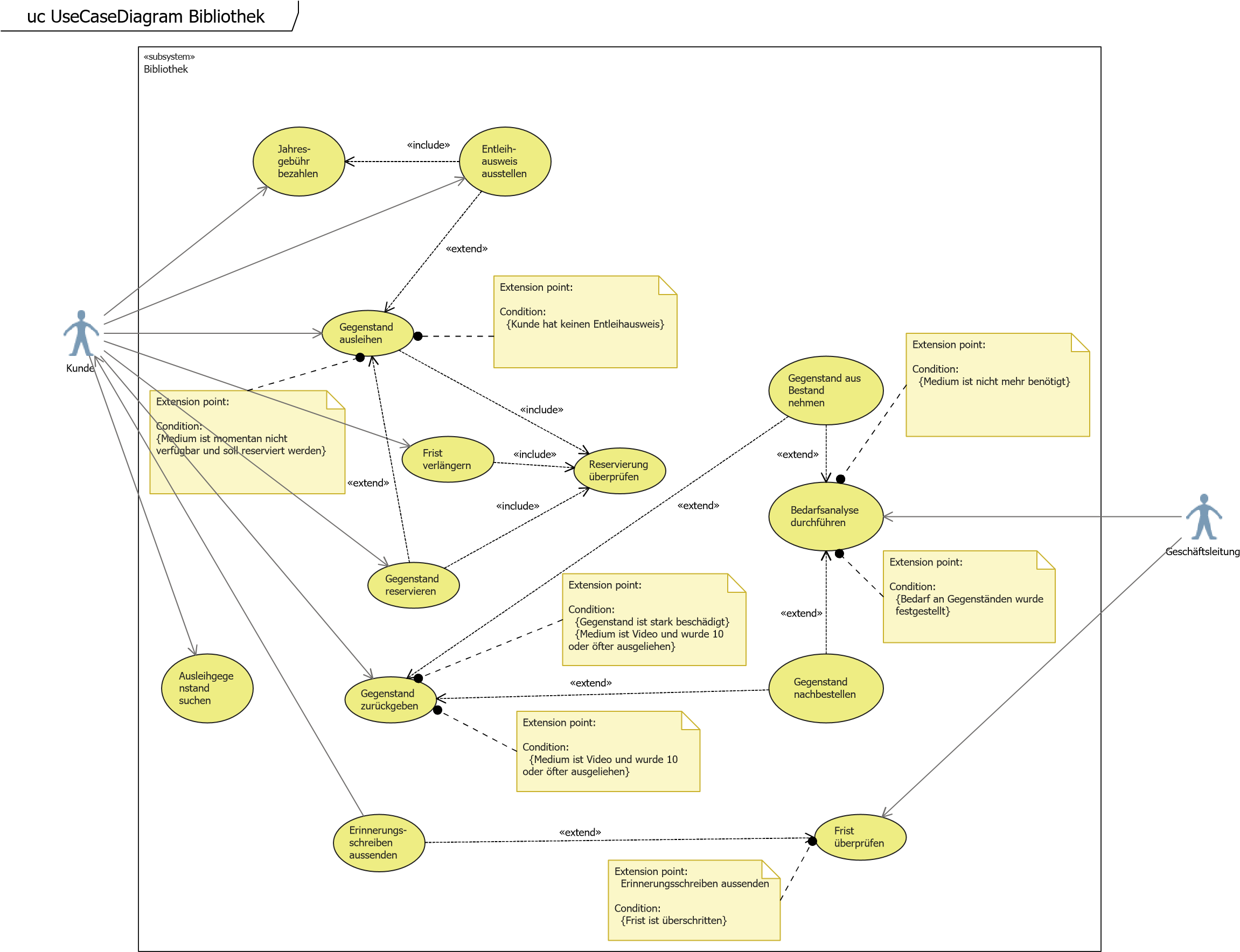
* Risikofaktoren können schnell erkannt werden.
* Bessere Arbeitsteilung anhand verschiedener Rollen
* Man kann sich auf mögliche Probleme konzentrieren
* Laufende Qualitätssicherung
* Prozess verbessert sich nach jeder Iteration

**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

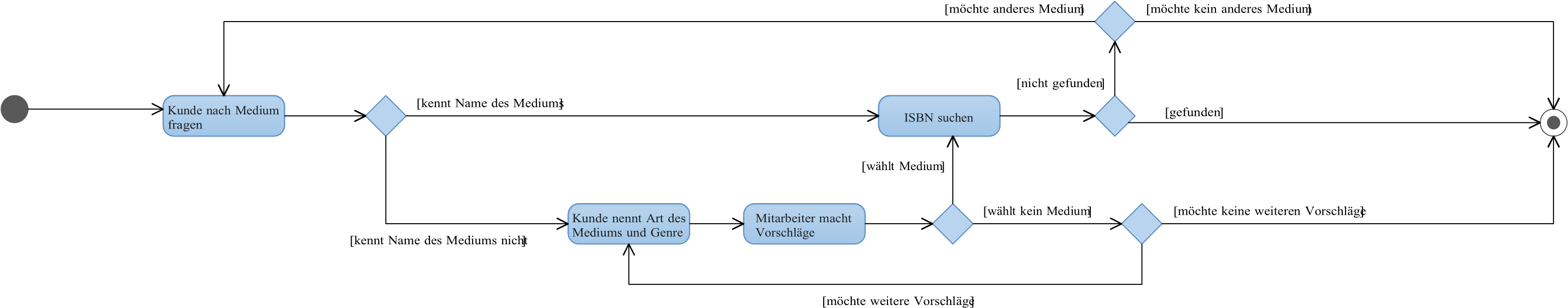
**Auslösendes Ereignis:**

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>