In diesem Kapitel:
- *Geschichte von RUP/ROP*
- *Inkrementelle und iterative Softwareentwicklung*
- *Startphase (inception phase)*
- *Entwurfsphase (elaboration)*
- *Konstruktionsphase (construction phase)*
- *Übergangsphase (transition)*
- *UML und RUP/ROP*

9

Die Unified Modelling Language UML und der Rational Unified Process RUP / Objectory

9.1. Ursprung

Der Rational Objectory Process zur Software Entwicklung hat im wesentlichen zwei Ursprünge. Der erste liegt im Objectoyr Prozess, der in Schweden durch Ivar Jacobson bei Objectory AB entwickelt und weltweit von zahlreichen Unternehmen übernommen und erfolgreich eingesetzt wurde. Schwerpunkt dieses Prozesses sind Use Cases gekoppelt mit einem objektorientierten Vorgehensmodell für das Softwaredesign. In den Grundzügen wurde dieses Vorgehen 1992 in einem Buch von Jacobson veröffentlicht. Der zweite liegt im Prozess zur Softwareentwicklung, der von mehreren Mitarbeitern der Firma Rational (u.a. Kruchten, Booch, Roye) entwickelt wurde. Schwerpunkte dieses Prozesses sind das iterative Vorgehen und die zentrale Stellung der Softwarearchitektur im Softwareentwicklungsprozess. Diese Vorgehensweise wurde verschiedentlich veröffentlicht. Die beiden Prozesse Objectory und Rational wurden 1996 (Ivar Jacobson war mittlerweise Mitarbeiter der Firma Rational) zum Rational-Objectory Prozess (im folgenden kurz ROP genannt) verschmolzen. Der Rational Objectory Prozess wird laufend von führenden Methodikern der Firma Rational (Grady Booch, Ivar Jacobson, Philippe Kruchten, Walter Royce und Jim Rumbaugh) weiterentwickelt. Die Version 5 des ROP wird neu als Unified Process bezeichnet, in Anlehnung an die Unified Modelling Language (UML), mit deren Hilfe die Methode arbeitet.

9.1.1. Was versteht ROP unter einem Prozess?

Definition (aus dem Einführungsmanual, welches als PDF von WWW.RATIONAL.COM herunter geladen werden kann):
Der Prozess stellt eine Menge geordneter Aktivitäten dar, mit denen im Softwareengineering die Erstellung oder Erweiterung eines Softwareproduktes erreicht werden soll.

Der Rational Objectory Prozess ist ein generischer Prozess zur objektorientierten Softwareerstellung beziehungsweise zur Erstellung von objektorientierter Software. Wie er an spezielle Prozesse angepasst werden kann wird später noch gezeigt werden.

Grundsätzlich unterscheidet ROP zwei Prozessstrukturen:
- Einmal die statische Struktur mit dem Schwerpunkt, *was* zu tun ist. Der Darstellung dieser Struktur werden wir uns als erstes widmen (Aktivitäten, Artefakte, Rollen)
- Dann die dynamische Struktur, die beschreibt, *wann* etwas getan werden muss und *wer* es tun sollte. In der dynamischen Struktur werden die Elemente der statischen Struktur in einem Zeitablauf auf Phasen und Iterationen verteilt, und es werden Meilensteine gesetzt,

in denen festgelegt wird, was am Ende der Phasen erreicht sein muss. Weiterhin werden die Aktivitäten der statischen Strukturen durch Aktivitäten Rollen zugeordnet werden

9.2. Summary und Vorschau

Im Folgenden werden die Grundbegriffe der dynamischen und der statischen Prozessstruktur beschrieben, und es wird dargestellt, wie diese beiden Strukturen im ROP in einer einzigen Struktur harmonisiert werden. Modelle sind ein wesentlicher Bestandteil im ROP. Wir werden nicht alle Teile des ROP im Detail besprechen, aber uns genauer mit den Use Cases, dem Design- und Implementierungs-Modell mit UML befassen und daraus Nutzerforderungen zur Coderezeugung herleiten.

Wir gehen nachher auf das Architektur-Modell des ROP ein beschreiben kurz Beiträge des ROP für

- Projektmanagement
- Qualitätssicherung
- Konfigurationsmanagement

Ein wesentlicher Punkt im ROP ist das Risikomanagement. Wir widmen uns diesem Thema an Hand von Beispielen und einigen generellen Punkten.
Da ROP ein generischer Prozess ist, muss er vor der Anwendung in einem konkreten Projekt erst angepasst werden. Wir werden kurz zusammen fassen WAS dazu zu tun ist.

9.3. Literatur

Der ROP wurde von Rational in Buchform veröffentlicht und in Form einer multimedia CD ROM (frei) verbreitet.

Die umfassende Dokumentation in Papierform umfasst folgende Handbücher, wobei ich damit rechne, dass diese Liste laufend ergänzt werden muss.

- *Rational Objectory Process - Introduction*
  Ein allgemeiner Überblick über den gesamten ROP
- *Rational Objectry Process - Process Manual*
  enthält eine detaillierte Beschreibung aller ROP Aktivitäten und deren Zusammenhang sowie die zur Durchführung wichtigen Rollen
- *Rational Objectory Process - Artifacts*
  Enthält eine Beschreibung der Eingabe- und Ausgabe-Produkte der ROP Aktivitäten
- *Rational Objectory Process - Modelling Guidelines*
  Dies ist ein Begleitmanual zum Prozessmanual mit Richtlinien für die Anwendung der UML-Elementarmethoden
- *Rational Objectory Process - Process Configuration*
  beschreibt, wie man den generischen ROP an bestimmte Projekttypen anpassen kann
- *Rational Objectory Process - Project Management*
  gibt praktische Richtlinien zur Durchführung des Projektmanagements im ROP
- *Rational Objectory Process - User Guide for Rational Rose*
  Beschreibt, wie man ROP mit *Rational Rose* (dem CASE Tool von Rational) nutzen kann

Zahlreiche Artikel und Demoversionen kann man von www.rational.com herunterladen oder als CD-ROM gratis beziehen (Demo Versionen, auf dem Server).

9.4. Inkrementelle und iterative Softwareentwicklung

Zuerst wollen wir einige der Begriffe genauer erläutern:

9.4.1. Die zeitliche Projektentwicklung aus Sicht des Managements (Phasen)

Das Management interessiert sich (in der Regel) vor allem für den zeitlichen und den finanziellen Rahmen der Projekte.

Zeitliche Aspekte werden in der Form von Phasen (sinnvoll) zusammen gefasst.

Finanzielle Aspekte werden in der Form einer Wirtschaftlichkeitsrechnung transparent gemacht.

Am Ende jeder Phase wird ein Meilenstein gesetzt, zu dem der Auftraggeber bestimmte Ergebnisse überprüfen kann und in der Regel auch eine detailliertere Planung der nächsten Phase, sowie ein Hochrechnen des Gesamtprojektes erwartet.

Die Phasen unterscheiden sich etwas, je nach dem zu Grunde liegenden Phasenmodell (linear, Wasserfall, Spiralmodell, mit oder ohne Prototyping).

Das lineare Modell mit den Phasen:

- Analyse
- Design
- Programmierung
- Test

Ist in der Regel zu weit von der Realität entfernt. Projekte lassen sich in der Regel nicht linear planen und abwickeln. Es gibt viele Gründe dafür:

- Die Benutzer ändern sich
- Die Technologie entwickelt sich weiter
- Die Firma kann auf Grund der wirtschaftlichen Entwicklung andere Schwerpunkte setzen
- ...

Der Begriff der Phase hat sich jedoch als sinnvoll erwiesen. Er muss nur neu interpretiert werden:

- er muss einem inkrementellen Vorgehen angepasst werden
- er muss ein iteratives Vorgehen ermöglichen

beziehungsweise:
der Begriff muss so modifiziert werden, dass ein inkrementelles und iteratives Vorgehensmodell möglich wird.
Wir werden die Uminterpretation gemäss dem ROP besprechen und anschliessend zeigen, wo und in welcher Form die klassischen Begriffe

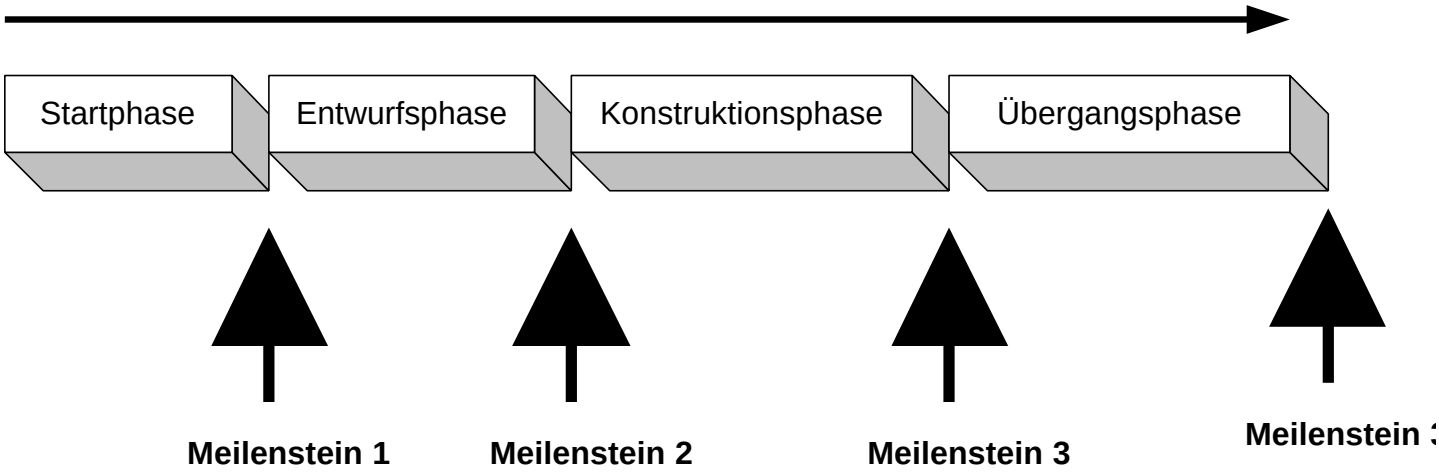
- Analyse
- Design
- Programmierung
- Test

wieder auftauchen.

ROP kennt vier Phasen zur Darstellung der *zeitlichen* Entwicklung eines Software Systems:

- Startphase (inception phase)

- Entwurfsphase (elaboration phase)
- Konstruktionsphase (construction phase)
- Übergangsphase (transition phase)



Am Ende jeder Phase steht ein Meilenstein. Zu diesem Zeitpunkt müssen bestimmte (pro Phase) vordefinierte Ergebnisse vorliegen.

9.4.1.1. Die Startphase (inception phase)

Die Startphase soll ein besseres Bild über das Gesamtprojekt liefern. Die einzelnen Informationen liegen aber erst in einer Form vor, dass entschieden werden kann ob das Projekt weitergeführt bezw. überhaupt in Angriff genommen werden soll.

Falls sich einzelne Anforderungen als zu ehrgeizig oder zu innovativ erweisen, kann in dieser Phase noch eine Korrektur der Projektziele vorgenommen werden.

Das setzt voraus, dass speziell die risikoreichen Projektteile genauer untersucht werden.

Ergebnisse am Ende der Startphase:

- Grobes Modell aller zu bearbeitenden Problembereiche (domains)
- Generelle Übersicht über alle wichtigen Nutzerforderungen innerhalb der Fachbereiche
- Grober Phasen- und Projektplan (Zeit, Meilensteine, personelle und finanzielle Resourcen)
- Überblick über die zu erwartenden Projektkosten und die Erfolgskriterien für das Projekt
- Erste Identifizierung der Risikofaktoren
- Allgemeine Übersicht über das Use Case Modell (10-20% fertig) mit Use Case Beschreibung
- Ein Glossar der wichtigsten Begriffe zur Vervollständigung zwischen Benutzer und der Informatik. Dieses stellt ein sehr wichtiges Ergebnis dar, da erfahrungsgemäss nur mit einem solchen Glossar eine sinnvolle Verständigung zwischen allen Projektbeteiligten erreicht werden kann

Je nach Projekt kann es sich als notwendig erweisen, in der Startphase bereits einen Prototypen zu erstellen. Dieser dient in der Regel dazu, bestimmte Systemeigenschaften (Performance, Benutzerschnittstelle) zu testen und besser planbar zu machen.

Der erste Prototyp ist in der Regel ein Wegwerfprodukt, wird also nie produktiv.
Ein weiterer Prototyp kann dann evolutionär zum eigentlichen Produkt weiter entwickelt werden.
Die Startphase kann sehr kurz sein, einige Tage oder wenige Wochen.

9.4.1.2. Die Entwurfsphase (Elaboration Phase)

In der Entwurfsphase ist der Anwendungsbereich zu analysieren. Wesentlich für diese Phase ist die Entwicklung eines tragfähigen Architekturmodelles, sowie einem detaillierten Projektplan.

Die Risikoanalyse, die in der Startphase bereits gestartet wurde, wird fortgesetzt und vertieft und Lösungsmöglichkeiten beziehungsweise worst case Szenarien ausgearbeitet.

9.4.1.2.1. Die wesentlichen Ergebnisse der Entwurfsphase:

- Use Case Modell (ca. 80% vollständig), für alle Bereiche, speziell die riskoreichen
- Softwarearchitektur, die tragfähig für das Projekt ist
- Verfeinerter Projektplan,
  der gegebene *Rahmenbedingungen* berücksichtigt (Hardware, Systemsoftware, Netzwerke, Datenbanken)
  der allfällig planbare *Iterationen* berücksichtigt
- Evtl. Ergebnisse von Tests mit Architekturprototypen und den Use Cases, die mit höchster Priorität zu implementieren sind
- Eine verfeinerte Risikoanalyse und Risikobewertung

9.4.1.2.2. Welche Risiken können typischerweise auftreten:

- Anforderungsrisiken : bauen wir das richtige System
- Risiken bei den Fähigkeiten : verfügen die Mitarbeiter über die benötigten Fähigkeiten
- Technologische Risiken : wie gut kennen Sie neue Techniken (OO, Java, WWW)
- Politische Risiken : wer hat welche Interessen innerhalb der Firma

9.4.1.3. Die Konstruktionsphase (Construction Phase)

In der Konstruktionsphase sind alle Software-Produkte iterativ fertig zu stellen, inklusive Tests.

Ausgangspunkt der Softwareerstellung ist eine detaillierte Softwarearchitektur, die in der Entwurfsphase erstellt wurde, sowie deren Abbildung auf eine vorhandene bezw. konzipierte Hardwareumgebung.

Die restlichen Use Cases müssen vervollständigt werden.

Am Ende der Phase muss entschieden werden, ob die Software dem Auftragbeger zum praktischen Einsatz übergeben werden kann.

Die wesentlichen Ergebnisse der Konstruktionsphase:

- Erstellte und getestete Software
- Benutzermanual
- Beschreibung des aktuellen Releases

9.4.1.4. Die Übergangsphase (Transition Phase)

In der Übergangsphase wird das neu entwickelte System beim Benutzer eingeführt. Bei kommerziellen Produkten findet bei ausgewählten Kunden zunächst eine Betatest-Periode statt (die Alphatest-Periode ist firmenintern).

Bewährt sich die Software, dann wird sie nach der Behebung der festgestellten Mängel als erstes Release (general available release) freigegeben.

Der Durchlauf eines Projektes durch alle Phasen wird als *Entwicklungszyklus* (development cycle) bezeichnet. Als Ergebnis entsteht eine erste Software Generation.

Die darauf folgenden Durchläufe werden als *Evolutionszyklen* (evalution cycles) bezeichnet.

Diese Phasensicht ist aber nur eine der möglichen und in RUP vorhandenen Sichten.
Die Sichtweise beschreibt den Projektablauf auf der Zeitachse. Allerdings denkt der Entwickler in der Regel anders. Wie man im System Engineering lernt, läuft beim Entwickler ein Problemlösungszyklus ab. Der Entwickler kennt also andere Phasen und Phasenplanungen. Diese widersprechen den zeitlichen Phasen nicht, sondern sie ergänzen diese.

Wie diese Problemlösungszyklen oder Phasen aussehen, wollen wir nun genauer anschauen.

9.4.2. Projektentwicklung aus der Sicht des Softwareentwicklers

Der zeitliche Ablauf des Projektes beschreibt die Detailaktivitäten des Entwicklers nur in ungenügendem Masse.

Der Entwickler denkt in Begriffen und Aktivitäten wie:

- Analyse
- Design
- Programmierung
- Test

wobei die einzelnen Aktivitäten jeweils iterativ (mehrfach) ausgeführt werden müssen.
Das Vorgehen lehnt sich an den Problemlösungszyklus gemäss dem Systemengineering an.

9.4.2.1. Ergebnisse

Das Ergebnis dieser Aktivitäten sind Produkte (Artefakte, artifacts). ROP unterscheidet zwischen *artifacts* und *deliverables*:

- *Deliverables* sind das Ergebnis einer Phase und für den Benutzer nutzbar
- *Artifacts* sind Ergebnisse, im Projektverlauf entstehen, aber nicht direkt für den Benutzer bestimmt sind.

Die Unterscheidung ist aber eher künstlich und wird auch nicht konsequent eingehalten. ROP verwendet daher artefacts auch für deliverables (als Synonym).
Zur Durchführung bestimmter Tätigkeiten sind bestimmte Fähigkeiten notwendig. Diese werden in Form von Rollen beschrieben.
Tätigkeiten (Aktivitäten), Produkte und Rollen werden im ROP unter dem Begriff *Prozessbereich* (process components) zusammen gefasst.

ROP kennt folgende sieben Prozessbereiche:

- Vier Hauptprozessbereiche
  1. Erfassung der Anforderungen (*requirement capture)*
  2. Analyse und Design *(analyse and design)*
  3. Implementierung (*implementation)*
  4. Test *(test)*
- Drei unterstützende Prozessbereiche
  1. Management
  2. Verteilung *(deployment)*
  3. Umgebung (*environment)*

Zu jedem Prozessbereich gehören:

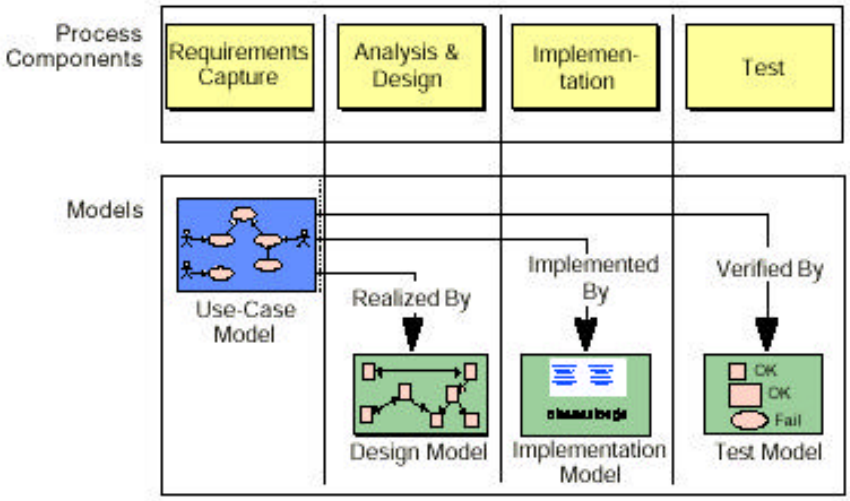
- Aktivitäten *(activities)*
- Rollen *(workers)*
- Produkte *(artifacts)*

9.4.2.2. Ergebnisse eines Prozessbereiches

Die wichtigsten Endprodukte eines Prozessbereiches sind die folgenden Modelle:

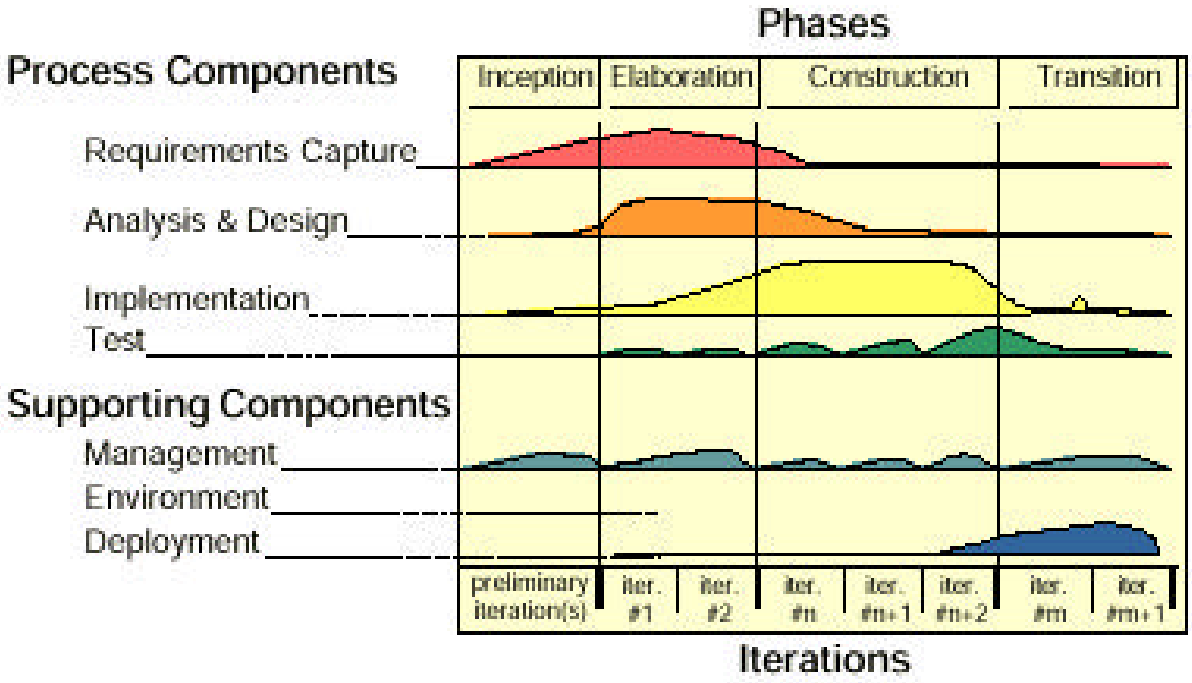
- Use Case Modell (Ergebnis der Erfassung der Anforderungen)
- Designmodell (Ergebnis der Analyse und des Designs)
- Implementierungsmodell (Ergebnis der Implementierung)
- Testmodell (Ergebnis der Tests)

Auszug aus Intro.PDF (Seite 18)



9.4.3. Integration der Projektentwicklung aus Sicht des Managements und der Entwickler

Gesamtübersicht gemäss Intro.PDF (Seite 13)

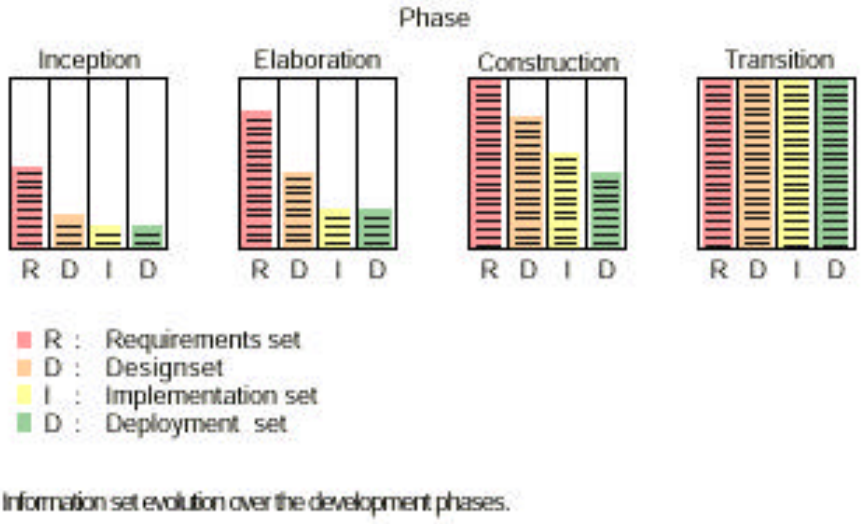


Die zeitliche Sicht wird auf der Horizontalen dargestellt, während die Sicht der Entwickler im Prozessbereich vertikal abgebildet wird.

Die Iterationen werden jeweils pro Phase aufgelistet. Die Schwerpunkte der einzelnen Phasen sind aus den farbigen Ausmalungen erkennbar.

Wesentlich an diesem Modell ist, dass auch zu einem späteren Zeitpunkt noch Anforderungen einfliessen können (im Sinne eines iterativen Vorgehens). Dies lässt sich wie folgt anschaulich darstellen (Fertigstellungs-Grad von Anforderungsanalyse, Design, Implementierung, Verteilung):

(Auszug aus Intro.PDF Seite 13)



Information set evolution over the development phases.

9.4.4. UML und Objectory (ROP/RUP)

UML, die Unified Modelling Language ist eine Beschreibungssprache, welche recht universell einsetzbar ist

RUP / ROP ist eine Methode, welche UML (aber auch zusätzlich andere Beschreibungsverfahren) als Beschreibung einsetzt.