Marknadsdriven kravhantering med Rational Unified Process

Examensarbete
Stina Åkesson
Salomeh Kiani

Handledare
Johan Natt och Dag, LTH
Björn Regnell, LTH
Gustav Bergman, Softhouse Consulting Öresund AB

Institutionen för telekommunikationssystem

LUNDS TEKNISKA HÖGSKOLA
Lunds universitet

CODEN:LUTEDX(TETS-5492)/1-120/(2003) & local 21

Abstract

In spite of the attention that research has paid market driven requirements engineering during the last ten years, there is still no comprehensive development process support. This thesis proposes a modification of the otherwise contract focused development process framework Rational Unified Process (RUP). The prime focus is enhancement of the elicitation process, adding a product lifecycle perspective and bringing forward the information gathering activities that will enhance the decision making process of release planning. New roles in RUP include Product Manager and Requirements Database Management. A new workflow is introduced with the Release Planning Workflow. Project Management Workflow and Requirements Workflow are modified, based on an analysis of RUP, establishing to what degree it is affected by being contract driven. The proposed framework (MRUP) has been developed during five iterations with subsequent feedback from software industry representatives and researchers.

Innehållsförteckning

Förord

Examensarbetet genomfördes på Softhouse Consulting Öresund AB och Institutionen för Telekommunikationssystem på LTH under sommaren och hösten 2003.

Först och främst vill vi tacka Softhouse Consulting Öresund AB som gjorde det möjligt att genomföra examensarbetet. Särskilt tack till Gustav Bergman. Tack för entusiastisk handledning från institutionen för Telekommunikationssystem: Johan Natt och Dag och Björn Regnell.

Vi är mycket tacksamma för det intresse vi mött och den tid och erfarenhet vi fått tillgång till vid utvärderingarna. Tack till Per Beremark, Appium AB; Thomas Hjelm; Peter Wennerborn och Staffan Persson, Softhouse Consulting Öresund AB; Micael Åkesson, Sony Ericsson samt Magnus Gerward, Ericsson Mobile Platforms.

Tack också till Magnus Höglund, Focal Point AB i Linköping för ett mycket givande och inspirerande studiebesök.

Tack Henrik Palm för skoningslös korrekturläsning.

Malmö, november 2003

Salomeh Kiani & Stina Åkesson

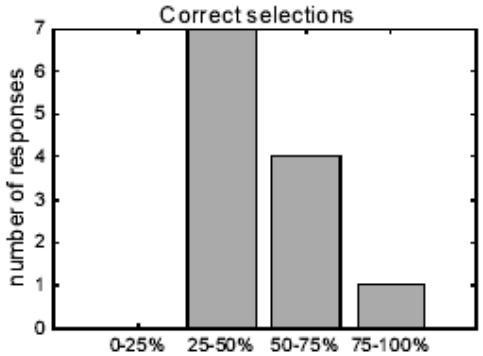
LUNDS TEKNISKA HÖGSKOLA
Lunds universitet

Kapitel 1

Inledning

Länge har programvaruindustrin utgått från att det normalt sett finns en specifik kund för programvaran och ett kontrakt att uppfylla (Potts, 1995, s 130). Det kan verka lite förvånande eftersom kommersiell programvara, utvecklad för en mer eller mindre öppen marknad knappast är en nyhet. De senaste tio åren har dock forskningen riktat mer uppmärksamhet åt problemen som den typen av utveckling är drabbad av[1]. Skillnaderna i arbetssätt är stora och det finns forskare som menar att många traditionella kravhanteringsmetoder helt enkelt är olämpliga i det marknadsdrivna perspektivet (Natt och Dag, 2002, s 17). Trots det är nästan alla metoder och processmodeller som finns inriktade på utvecklingen av programvara för en kunds räkning (Carmel & Becker, 1995, s 50, Karlsson, 2002, s 5). Företag som utvecklar programvara kommersiellt befinner sig alltså i en situation där dagens processtöd är mindre väl anpassat för deras behov.

Vid s.k. kontraktsutveckling samlar det utvecklande företaget först in vilka krav som finns på produkten, drar gränser för vad systemet ska göra och prioriterar vilka krav som är viktigast. Kunden själv godkänner kravspecifikationen, som fungerar som kontrakt mellan de två organisationerna. Kunden godkänner senare den färdiga produkten. Inom den s.k. marknadsdrivna utvecklingen finns inga kontrakt och problematiken handlar istället mer om att utveckla ”rätt” produkt, d v s en produkt som är framgångsrik på marknaden och som når ut på marknaden i rätt tid. En undersökning av Regnell et al. (se figur 1.1) visar att i 7 av 12 fall bedömde företagen i efterhand att endast för 25-50 % av kraven hade rätt beslut fattats om att inkludera eller inte inkludera det enskilda kravet (Regnell et al. 2003). Det visar på hur svårt det är att fatta beslut om produktinnehåll i en föränderlig värld. Till stor del är det viktiga kundgruppers behov som styr, men också konkurrenters agerande på gemensamma marknader och timingfrågor. Behoven kan vara behov som faktiskt redan finns, eller så kan de skapas av företaget eftersom tillgång till en viss teknologi kan bana väg för nya produkter eller produktegenskaper. Vem hade t ex behov av SMS innan det fanns på marknaden? Att löpande förse sig med goda kunskaper om marknaderna och att ha beredskap för förändringar är essentiellt för att fatta bra beslut om produktinnehållet. Långsiktigt sett är det den kommersiella framgången som räknas.



Figur 1.1 *Distribution av andel korrekt fattade beslut att ta med/inte ta med ett krav i en produkt. Ur Regnell et al. (2003).*

[1] Se Potts, 1995; Yeh, 1992; Lubars, Potts & Richter, 1993; Natt och Dag, 2002, Regnell et. al, 1998.

Rational Unified Process (RUP) är ett processramverk för utveckling av programvara. Den är den ett stort antal länkade HTML-sidor och dokumentmallar med tillhörande förklaringar. Arbetsgången för programvaruutvecklingen beskrivs i ett antal arbetsflöden, (t ex för kravhantering) där olika aktiviteter ska genomföras i en viss ordning och dokument ska produceras. RUP blir ofta kritiserat för att vara ett mycket omfattande processramverk, men Rational som äger RUP rekommenderar att det utvecklande företaget själv eller via konsulter väljer ut de delar som organisationen faktiskt behöver.

Rational själva menar också att RUP använder sig av praxis från programvaruindustrin som bevisats fungera i alla typer av programvaruprojekt. Det rimmar dock lite illa med vittnesmålen om bristande processtöd för kommersiell programvaruutveckling. I realiteten använder en del företag en kompromisslösning där marknadsavdelningen eller en produktledare agerar beställare av programvaran. Anledningen till att detta examensarbete genomförts är att konsulter på Softhouse Consulting Öresund AB erfarit otillräcklighet med hur RUP fungerar när den används marknadsdrivet och att RUP borde kunna anpassas till att bättre fylla behovet av stöd för marknadsdriven utveckling.

Kapitel 2

Syfte

Det huvudsakliga syftet för examensarbetet var att skapa ett förslag till modifiering av RUPs kravhantering som är bättre lämpad för kommersiell utveckling än den nuvarande. Den modifierade versionen benämns i rapporten som MRUP – Market-driven Rational Unified Process. Arbetet utfördes på Softhouse Consulting Öresund AB i Malmö.

För att ha en bra utgångspunkt för den nya versionen krävdes ett underlag för vad som är mindre bra för marknadsdriven utveckling med RUP som den ser ut idag. Forskning inom området har försökt sammanställa typiska egenskaper för marknadsdriven och kontraktsdriven utveckling. Dessa egenskaper användes för att analysera RUP. Målet var att identifiera egenskaper som är kontraktsinriktade och som gör det svårt att använda RUP marknadsdrivet utan anpassningar.

Ambitionsnivån för examensarbetet var att göra ett förslag till förändring av de delar av RUP som berör kravhantering. Inspiration till den nya modellen kom från aktuell forskning inom marknadsdriven kravhantering, produktteori och software engineering. Inom tidsrymden för ett examensarbete fanns det bara utrymme för att göra en ganska schematisk modell. Fokus låg på att hitta relevanta arbetsgångar och ungefärligt innehåll i varje processteg. Implementering, integrering i RUP och beskrivning av processen på detaljnivå ligger utanför omfånget av examensarbetet, men går troligen relativt smidigt för en rutinerad RUP-konsult att genomföra.

Målgruppen för denna *rapport* är en sistaårsstudent inom civilingenjörsutbildningen och därför utformad för att de ska förstå problematiken. Examensarbetets *innehåll* är riktat till civilingenjörer verksamma inom programvaruutveckling.

Utveckling av processramverk är ett brett arbete vilket också är tydligt i den här rapporten. För att ge en teoretisk grund för läsaren finns en genomgång av relevanta begrepp i kapitel 3, med inriktning på bredd snarare än djup. Först görs en genomgång av definitioner för krav, därefter motsvarande för kravhantering. De delar av kravhanteringen som RUP i nuläget inte behandlar särskilt ingående får lite fylligare beskrivningar. De större skillnaderna mellan marknadsdriven och kontraktsdriven utveckling redovisas. Därefter redovisas översiktligt de marknadsteoretiska begrepp som kommer att beröras av MRUP och avslutningsvis beskrivs hur RUP är uppbyggt och vad som ingår i tre för kravhantering centrala arbetsflöden.

Kapitel 4 behandlar metoden för analysen av RUP och hur konstruktionen av ett förslag till en marknadsdriven RUP gått till.

I kapitel 5 redovisas resultatet av analysen, dels som en övergripande kvantifierad redovisning, dels med en mer kvalitativ genomgång. Den kvantifierade redovisningen visar hur mycket hänvisningar till kontraktsinriktade begrepp som arbetsflödena använder sig av. Den kvalitativa redovisningen går sedan lite djupare i analysen och tittar på hur allvarligt det är med de kontraktsinriktade egenskaperna om man försöker använda RUP marknadsdrivet.

I kapitel 6 redogörs för hur MRUP vuxit fram och vilka insikter som vunnits på vägen. Också själva förslaget återges i detta kapitel. Två omarbetade dokument: *Vision* och *Business Case*, finns i Appendix A.

Sammanfattande slutsatser och diskussion följer i kapitel 7 och slutligen rekommendationer för fortsatt arbete i kapitel 8.

Kapitel 3

Teori

3.1 Krav

Tyvärr används inte begreppet krav på samma sätt i hela programvaruindustrin. Detta kapitel syftar till att ge en presentation av olika definitioner och avslutas med hur begreppet kommer att användas i rapporten.

Det finns två huvudfokus i olika definitioner av *krav* (*requirement*): en användarorienterad inriktning och en som fokuserar på systemet. Traditionell kravhanteringslitteratur lägger fokus på hur systemet är utformat, vilket också Phillippe Kruchten, en av huvudarkitekterna för RUP gör i sin introduktionsbok till RUP[2]:

> ”ett villkor eller en egenskap som systemet måste uppfylla” (Kruchten, 2000, p 158).

I den ofta citerade[3] IEEEs[4] definition ingår bägge synsätten och det är också till denna som RUP hänvisar.

> ”1. A condition or capability needed by a user to solve a problem or achieve an objective.
> 2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.
> 3. A documented representation of a condition or capability as in 1 or 2.” (IEEE (1990) Standard Glossary of Software Engineering Terminology, ANSI/IEEE Standard 610.12, New York, USA1990).

I den första punkten av IEEEs definition ingår användaren som har ett behov av en funktion eller tillstånd för att antingen lösa ett problem *eller* uppnå ett visst syfte. För de mer användarcentrerade kravdefinitionerna är detta en generös formulering eftersom det inbegriper utveckling av programvara som inte bara ”löser ett problem” (vilket är ett vanligt synsätt), utan också kan inbegripa utveckling av produkter för t ex underhållningsindustrin.

Den andra punkten fokuserar på mer formaliserade krav som systemet eller dess komponenter måste leva upp till och som kan spåras till någon typ av formell dokumentation som kontrakt, standarder, specifikationer eller dylikt. Valfriheten för kraven är här begränsad och för kontraktsbaserad utveckling är detta den centrala punkten. Det bör poängteras att t ex inom telekomindustrin är en icke föraktfull del av kraven standardiseringskrav från operatörerna.

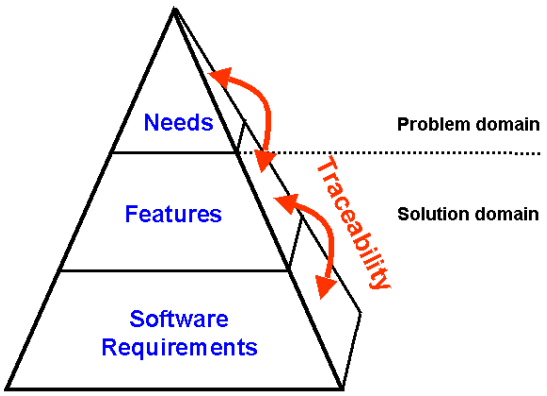
Den tredje punkten innebär att även om kravet är ett *behov* som registrerats från användaren, vilket i normalt språkbruk inte har en lika obligatorisk klang som *krav*, så definieras det som ett krav i sin dokumenterade form. Enligt IEEEs terminologi, är ett behov ett krav redan innan det dokumenterats och ses då som ”råkrav” (*raw requirement)* (IEEE Standard 1233-1998. IEEE guide for developing system requirements

[2] Se också Kotonya & Sommerville, 1998, s 3, som också är systemfokuserad.
[3] Se t ex Leffingwell & Widrig, 2000, s 15.
[4] IEEE (the Institute of Electrical and Electronics Engineers, Inc.) är en ideell organisation som utarbetar tekniska standards, publicerar teknisk litteratur och arrangerar konferenser. IEEE verkar inom elektro- och datateknik, biomedicin mm.

specifications). På så sätt skulle det inte bli någon skillnad på behov och krav, men det finns en poäng i att skilja på sådant som har att göra med lösningen och sådant som är kopplat till *varför* lösningen finns. Leffingwell och Widrig gör just denna distinktion och definierar *feature*, eller funktionellt högnivåkrav som vi väljer att kalla det på svenska, som

> ”a service that the system provides to fulfill one or more stakeholder needs” (Leffingwell & Widrig, 2000, s 20).

Relationen till behov illustreras i figur 3.1.1 där alltså en viss feature kan spåras till ett eller flera behov.



Figur 3.1.1 *Översikt över relationen behov, features och programvarukrav. Ur Leffingwell & Widrig (2000).*

Ett problem med kravbegreppet är att det både handlar om önskemål om ett systems utformning och om krav som måste uppfyllas. Regnell poängterar beslutsprocessen som finns med då kraven specificeras:

> “We can see that the word requirement can mean both a desired property and an obligatory property. This reflects the decision process inherent to requirements engineering; we need not only to find all desires, we also have to decide which of the, possibly conflicting, desires to be implemented” (Regnell, 1999, s 12).

Det finns också stora skillnader i hur krav uttrycks. Ibland är abstraktionsnivån hög, vilket kan vara fallet då det handlar om användarkrav eller ett kontrakt som förhandlas fram (Davis, 1993, Sommerville, 2001, s 98) eller då urvalet av features till en ny produktrelease väljs ut för implementation. Inför implementationen görs en mer detaljerad systemdefinition som antingen utgör själva kontraktet med en specifik kund eller som kan användas som kommunikationsmedel inom det utvecklande företaget[5]. Lauesen menar att användarkrav helt enkelt syftar på krav på produktnivå och ska kunna förstås av användare, medan systemkrav är mer tekniska till sin natur och riktade mot utvecklare (Lauesen, 2002, s 22). Leffingwell och Widrig gör relationen tydlig i figur 3.1.1 genom att bara kalla de tekniska specifikationerna för *software requirements*. Problem uppstår när termen ”krav” används utan att specificera om det rör sig om systemkrav eller användarkrav. Sommervilles rekommenderade lösning för att skilja på abstraktionsnivåerna är att helt enkelt separera användarkrav från systemkrav.

Sommervilles synsätt har sina poänger då det handlar om utveckling som syftar till att producera en produkt som ska bli framgångsrik på marknaden. Beslut måste fattas om vad som ska ingå i produkten. Den ska ha ”rätt” egenskaper och nå marknaden så fort som

[5] Se bland annat Sommerville, 2001, s 98.

möjligt. Litteratur som koncentrerar sig på marknadsaspekterna inom programvaruutveckling fokuserar gärna på användaraspekter vid definitionen av krav, poängterar att om det handlar om en feature, funktion eller annan egenskap hos systemet så ska det gå att iaktta utanför systemet. Ofta citeras Davis[6]:

> "A user need or a necessary feature, function, or attribute of a system that can be sensed from a position external to that system." (Davis, 1993, s 371)

Den marknadsinriktade Karlsson skiljer explicit på *behov* och *krav*. Kunder har behov och krav innebär översättning av behovet till en viss produkt eller tjänst (Karlsson, 2002, s 43). Beslutsprocessen kan då initialt begränsas till beslut om vilka behov produkten ska uppfylla. De mer tekniska aktiviteterna kan sedan inriktas på hur behovet ska uppfyllas vid formuleringen av krav. Problemet med Karlsson är att han i några artiklar använder *behov* för att beteckna behov och krav[7].

En begränsande faktor i mer användarcentrerade kravdefinitioner är användaren som intressent, d v s någon som har materiellt intresse i programvaran. Om inte användaren är slutkund har t ex den som är köparen av systemet behov, liksom andra intressenter som tekniska administratörer, underhållare och utvecklare av systemet, vilket också Karlsson noterar (Karlsson, 1998, s 2).

Vid utformning av ett nytt processramverk finns det skäl att använda en del av spännvidden för begreppet krav. Definitionen från IEEE utgör utgångspunkten för rapporten eftersom det är den som RUP använder sig av och som analyseras. Vid utvecklingen av ett nytt processramverk kommer dock begreppet snarare att ansluta sig till Davis definition, och då avse krav på hög abstraktionsnivå, eftersom de nya processerna gäller just kravhanteringens initiala skeden. Det är dock inget som utesluts av IEEEs definition och bör inte innebära något problem vid en integrering av prototypen i resterande RUP. Kraven kommer att benämnas *högnivåkrav, feature* eller *användarkrav*. Syftet till att en feature finns benämns *behov* eller *rationale*. Det finns en viss risk att begreppen blir inkonsekventa med resten av RUP, eftersom vissa delar kvarstår oförändrade. För att få en komplett marknadsdriven RUP måste naturligtvis resten av RUP få en genomgång för att uppnå konsekvens i begreppsbildningen. Troligare är dock att just kravdefinitionen inte har någon större praktisk betydelse eftersom de ”orörda” delarna hanterar de tekniska detaljspecifikationerna av högnivåkraven. Det går då mycket väl att låta RUP använda IEEEs definition, eftersom stödet för marknadsdriven utveckling saknades i de initiala aktiviteterna. För att göra rapporten tydligare används termen *systemkrav* för att markera att det rör sig om krav på detaljnivå.

3.1.1 Kravtyper

I allmänhet drar kravhanteringslitteraturen en skiljelinje mellan funktionella och icke-funktionella krav[8]. Med funktionella krav menas de tjänster som systemet kan utföra. De beskrivs med alla olika kombinationer av indata och utdata som ska kunna hanteras av systemet (Karlsson, 1996, s 12), ofta med hjälp av användningsfall.

[6] Se t ex Carlshamre, 2001, s 43, Karlsson, 1998, s2.
[7] Det gäller framför allt Karlsson, 2002.
[8] Leffingwell & Widrig, 2000; Lauesen, 2002; Kotonya & Sommerville, 1998; Karlsson, 1996.

Det finns däremot lite olika varianter på vad som ingår i de icke-funktionella kraven som ibland också benämns kvalitetskrav för att markera att de faktiskt måste fungera likaväl som funktionella krav (Lauesen, 2002, s 217). Huvudsakligen hanterar de icke-funktionella kraven:

- Användbarhet (*usability*)
- Tillförlitlighet (*reliability*)
- Prestanda (*performance*)
- Modifierbarhet (*supportability*)

(Grady, 1992; Leffingwell & Widrig, 2000, s 238)

Andra källor nämner icke-funktionella krav som tillgänglighet, säkerhet, portabilitet (IEEE 830) och hur väl systemet uppfyller standarder (ISO 9126). Lauesen (2002, s 220-223) har gjort en bra sammanställning över kvalitetskrav från tre källor, som han menar kan användas som en checklista vid utveckling för att se till att nyckelfrågor för det aktuella systemet tagits om hand.

3.2 Kravhantering (Requirements Engineering och Requirements Management)

Det svenska uttrycket *kravhantering* inbegriper de två engelska termerna *Requirements Engineering* (RE) och *Requirements Management* (RM). Man kan ana att det rör sig om flera relaterade verksamheter som samlas under samma begreppstak. Eftersom definitionerna ibland är överlappande och ibland skilda åt görs en genomgång av begreppen. Därefter definieras vad som avses med kravhantering i detta arbete och vilka delar som kommer att hanteras.

Huvudsakligen inriktades examensarbetet på att hantera processer som har med RE att göra. Regnell menar att resultatet av RE-processerna är kravspecifikationen och ringar på ett bra sätt in målet för vad RE bör åstadkomma:

> ”In particular we want to create a solid basis for:
>
> - planning and cost estimation
> - design and architectural decisions,
> - verification and validation.” (Regnell, 1999, s 10)

När det gäller RE finns det dock variationer för var fokus läggs: på kravinsamlingen och interaktionen med användare, eller mer på dokumentationen av krav. En ganska allmän definition hos Kotonya & Sommerville inkluderar:

> ”... all of the activities involved in discovering, documenting, and maintaining a set of requirements for a computer-based system.” (Kotonya, Sommerville, 1998, s 8).

Mer specifika med att det handlar om en social och kognitiv process är Loucopoulos & Karakostas som definierar RE som:

> ”… a systematic process of developing requirements through an iterative co-operative process of analyzing the problem, documenting the resulting observations in a variety of representations formats, and checking the accuracy of the understanding gained.” (Loucopoulos and Karakostas, 1995, s. 13)

En viktig poäng hos denna definition är att processen är iterativ, så de tre huvudsakliga delprocesserna kravinsamling (*elicitation*), specifikation (*specification*) och validering (*validation*) inte pågår som skilda ordnade aktiviteter. Gemensamt för många definitioner är att de tar fasta på dessa tre processer[9]. Att kraven till viss del handlar om identifiering av kundens och användarens behov är tydligt hos Hsia (1993) som definierar RE som:

> ”… all activities which are related to
>
> - identification and documentation of customer and user needs
> - creation of a document that describes the external behavior and the associated constraints that will satisfy those needs
> - analysis and validation of the requirements document to ensure consistency, completeness and feasibility, and
> - evolution of needs.” (Hsia et al., 1993, s 75)

[9] Se också Kotonya & Sommerville, 1998; Pohl, 1996; Macaulay, 1996; Karlsson, 1998; Karlsson, 1996

I Karlsson (1998, s 17-18) kategorisering finns möjligheten att ”uppfinna” behov[10] i den aktivitet som motsvaras Hsias första punkt. Särskilt gäller det om utvecklingen är teknikstyrd, d v s nya tekniska möjligheter kan innebära affärsmöjligheter för företaget, men behoven måste skapas hos marknaden.

Det råder viss oenighet om vad som ingår i RM. I en del definitioner ingår även processerna som hanterar RE:

> ”… a systematic approach to eliciting, organizing, and documenting the requirements of the system, and a process that establishes and maintains agreement between the customer and the project team on the changing requirements of the system” (Leffingwell & Widrig, 2000, s 16).

Det som tillkommer är processer för att genomföra ändringar i kravmängden på ett systematiskt sätt, vilket inbegriper bl a konsekvensanalys för systemet. Ändringshantering ingår i de flesta definitionerna av RM, men för att röra till begreppsbilden lite omfattar begreppet RE ibland även ändringshanteringen[11]. Ett problem med Leffingwell & Widrigs definition är att förändringar i kravmängden måste godkännas av en specifik kund. Ett annat problem är begränsningen till ett projekts livscykel istället för en produkts. Tyvärr är det denna definition som RUP använder sig av (se Concept: Requirements Management i RUP).

En annan variant är att RE hanterar själva kravanalysen medan RM enbart hanterar förändringar i kravmängden, som hos Kotonya & Sommerville. De tre huvudprocesser som RM då bör hantera är:

- förändringar i överenskomna krav (inom och mellan organisationer)
- relationer mellan krav
- beroenden mellan kravdokumentet och andra dokument i utvecklingsprocessen[12].

Fördelen med denna variant är att det inte finns några restriktioner i livscykeln eller huruvida kraven hör hemma i ett marknadsdrivet eller kontraktsdrivet arbetssätt.

I ovanstående definitioner saknas processer för prioritering av krav och releaseplanering, något som Carlshamre (2001, s 58) menar är specifikt för marknadsdriven utveckling. Karlsson (1998, s 17-18) ringar också in prioritering som en av huvudaktiviteterna för marknadsdriven RE. Det är denna beslutsprocess vid kravspecificeringen som Regnell syftade på (se kapitel 3.1).

I det svenska begreppet kravhantering undviks begreppsförvirringen delvis eftersom även ändringshanteringen ingår. Den definition för kravhantering som används i examensarbetet är baserad på Hsia men påbyggd med aktiviteter för prioritering, kostnadsuppskattning, releaseplanering och ändringshantering. Med *kravhantering* avses alla aktiviteter som är relaterade till

- identifiering och dokumentering av kund- och användarbehov (eller annan intressent)

[10] Termen ”invented requierements” är lånad från Potts (1995).
[11] Se Karlsson, 1998, s 18.
[12] Kotonya & Sommerville, 1998, s 114.

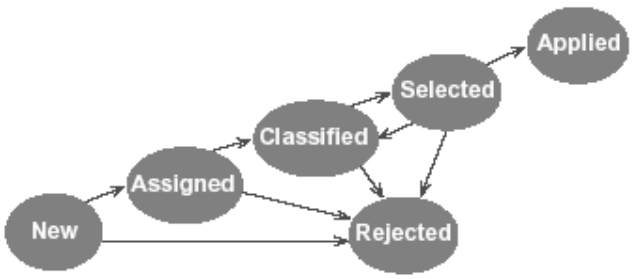
- skapande av ett dokument som beskriver funktionaliteten och de associerade begränsningar som uppfyller dessa behov
- prioritering och kostnadsuppskattning av krav
- releaseplanering
- analys och validering av kravdokumentet för att garantera att dokumentationen är konsistent, komplett och kraven spårbara och genomförbara
- behovens evolution
- förändringar hos kraven under hela systemets livscykel

I examensarbetet behandlas främst de fyra första aktiviteterna.

3.2.1 Dokumentering av krav

Även om en del mindre företag använder sig av textdokument för att dokumentera krav, är databasen vanligare. Fördelarna är att det går att få fram information om delar av kravmängden som har gemensamma egenskaper (t ex kraven i pågående release som ej är implementerade), att det går att navigera mellan krav som är relaterade, att uppdateringar inte kompliceras av fildelning och att hantering av spårbarhet, ändringar och beroenden blir smidigare (Kotonya & Sommerville, 1998, s 119). En kravdatabas kommer att innehålla kraven från en mängd releaser, deras attribut, spårbarhetsinformation, modeller och andra dokument som hör till kravet, samt ändringsbegäranden.

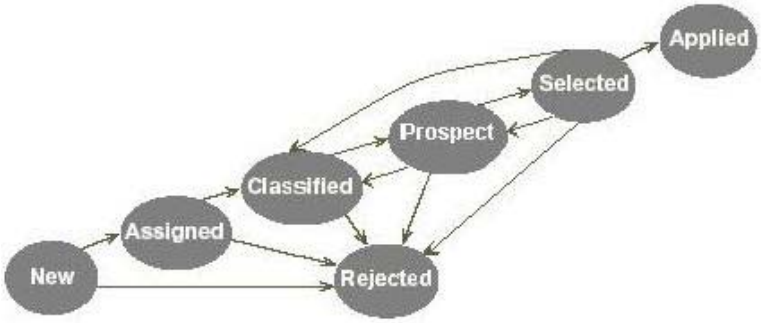
Krav beskrivs med hjälp av attribut som kan skilja sig mycket åt mellan olika organisationer och projekt. Litteraturen ger ganska olika förslag på attribut och det gäller att hitta en relevant uppsättning, eftersom administrationen av krav riskerar att bli en flaskhals om den blir för omfattande (Carlshamre & Regnell, 2000). Många böcker om kravhantering och programvaruutveckling menar att kraven utvecklas med tiden och att förändringar måste tas om hand på ett strukturerat sätt. Det är i och för sig riktigt, men få författare gör en explicit beskrivning av kravens livscykel. Arbetsinsatsen som behöver göras i olika tillstånd hos kravet kan göras mer kontrollerbar eftersom olika attribut kommer att vara aktuella i olika delar av kravets livstid. Livscykelbeskrivning via statusattribut är också smidigt att använda för att lämna statusrapporter från kravdatabasen med deluppsättningar av krav som har en viss status. Både Carlshamre och Regnell utgår från det i sina respektive modeller RDEM och REPEAT för hantering av inkommande krav (se Carlshamre & Regnell, 2000, för en jämförelse). Modellerna är ganska lika, men Regnells modell (se figur 3.2.1.1) har en något finmaskigare tillståndsmaskin med pedagogisk namngivning. Dessutom finns det ett tillstånd för bortsorterade krav, vilket inte Carlshamres modell har.



Figur 3.2.1.1 *Livscykel för krav i REPEAT-processen (ur Carlshamre & Regnell, 2000).*

New. Tillståndet när kravet först läggs in i databasen.
Assigned. En expert har tilldelats kravet för att närmare undersöka kravet och sätta ett antal attribut.
Classified. Experten har gjort en grov tidsuppskattning och bedömning av hur stora delar av koden som är berörd, samt eventuellt kommenterat och gett idéer till implementation.
Selected. Kravet kommer att implementeras till nästa release. De är då sorterade i prioritetsordning dels på en ”måste-ha”-lista och dels en ”önskelista”. De är detaljerat specificerade och har detaljerade kostnadsattribut och kodpåverkansattribut. Om ett krav i tillståndet *selected* plockas ur releasen, blir det antingen *classified* eller *rejected*.
Applied. Sluttillstånd. Kravet är implementerat och verifierat.
Rejected. Sluttillstånd. Kravet är utsorterat p g a det är en dubblett, redan implementerat eller inte ingår i produktens långsiktiga strategi.

I examensarbetet kommer en utvidgning av REPEATs livscykel att användas, se figur 3.2.1.2.



Figur 3.2.1.2 *Livscykel för krav i MRUP.*

Ett nytt tillstånd har införts: *Prospect*, som nås då kravet valts ut som kandidat till en release. Några attribut får då noggrannare värden och ytterligare attribut sätts. Om kravet väljs ut till releasen hamnar det i tillståndet *Selected*, men kan också gå tillbaka till *Classified* eller avfärdas helt och bli *Rejected*.

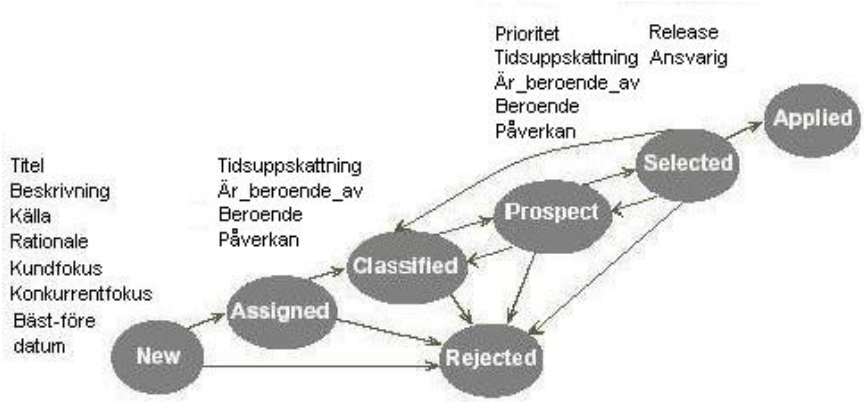
Eftersom varje organisation till slut måste avgöra vilken attributuppsättning som ska användas finns i tabell 3.1.1.1 en översikt över attribut hämtade ur Kotonya & Sommerville (1998), Leffingwell & Widrig (2000), Karlsson (1996, 2002), Alexander & Stevens (2002) och Regnell et al. (1998).

Attribut	**Beskrivning**	**Kategoriuppsättningar**
Identifikation	Unik beteckning. Kan numreras efter kapitelindelning, t ex 4.2.6 (kräver stabil indelning), eller databasen kan tilldela unika nummer. Även symbolisk identifiering kan göras: EFF-1, EFF-2 etc.	
Beskrivning	Beskrivning av kravet i naturligt språk eller med grafisk beskrivning.	

Rationale	Anledningen till att kravet finns. Kan vara ett övergripande behov från kundsegment eller ett dokument, t ex standardiseringsdokument. Kan benämnas reason eller orsak.	
Status	Tillståndet kravet befinner sig i under sin livscykel.	1. New-Assigned-Classified-Prospect-Selected-Applied-Rejected 2. Proposed-Approved-Incorporated
Värde/Prioritet		Kritiskt-Normalt-Användbart Sensationellt-Normalt-Förväntat Low-Medium-High Skala, t ex 1-5
Release	Releasen som kravet ska ingå i.	
Tidsåtgång (*effort, cost estimate)*	Hur lång tid kravet tar att implementera.	Persontimmar, -veckor, eller –månader Mindre än 1 dag-5 dagar-5 veckor-3 månader- mer än 3 månader Antal kodrader Antal funktionspunkter
Är_beroende_av (*Is_dependent_on*)	De krav som detta krav är beroende av.	
Beroende (*Dependants*)	De andra krav som är beroende av detta och som berörs vid ändringar.	
Påverkan (*impact*)	Hur många komponenter som påverkas.	En komponent-fåtal-färre än hälften-mer än hälften-nästan alla
Stabilitet	Sannolikhet att kravet kommer att ändras.	
Risk	Sannolikheten att kravet kommer att orsaka problem, t ex i form av fördröjningar, kostnadsökningar eller att det försvinner.	High-Medium-Low
Ansvarig	Ansvarig för design och implementering (projekt eller person).	
Datum_inlagt	När kravet lades in.	
Datum_ändring	Senaste ändringen av kravet.	
Bäst-före-datum	Det senaste datum då ett beslut om kravet bör ha tagits.	
Källor	Ursprunget till kravet: dokument (särskilt för standardiseringskrav), kunder, utvecklare.	
Kundfokus	Kundgrupper som kravet ska tillfredsställa (ej aktuellt för alla krav).	
Konkurrentfokus	Konkurrenter som kravet fokuserar på (ej aktuellt för alla krav).	
Verifiering	Hur kravet ska verifieras: testmetoder och testplan för kravet.	
Modellänkar	Länk till modeller och diagram.	
Kommentarer	Ytterligare användbar information.	
Nyckelord	Används för att gruppera krav.	

Tabell 3.2.1.1 *Sammanställning av kravattribut.*

Några attribut är mer centrala än andra för en ny RUP-modell, se figur 3.2.1.3.

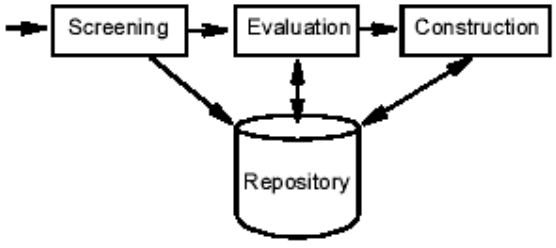


Figur 3.2.1.3 *Anpassad livscykelmodell kopplad till vilka attribut som specificeras/förändras i varje tillstånd.*

Enligt figuren återkommer samma attribut på flera ställen i livscykeln, särskilt attributen för tidsuppskattning och beroenden. Denna typ av information kan vara kostsam att ta fram och därför görs den olika noggrant i olika steg.

3.2.2 Behovsanalys

Detta avsnitt hade lika gärna kunnat benämnas ”kravinsamling”. En del av poängen med att tala om behovsanalys istället är att varje implementerat högnivåkrav bör motiveras av ett bakomliggande behov hos kunden och att detta synsätt är centralt för ett marknadsmässigt tänkande. Detta benämns ofta med rationale eller orsak (reason) vid beskrivningen av ett krav. Det finns knappt en bok om kravhantering som inte inleder med att dåliga krav leder till ökade kostnader, missade deadlines, missnöjda kunder och överarbetade anställda. Ibland handlar det om rena missuppfattningar om vad behovet egentligen var. Exempelvis kan det vara ett problem när krav som kommer till ett företag via supportavdelningen är beskrivna som lösningsförslag utan att det bakomliggande behovet är angivet. Alternativa, billigare lösningar då kanske förbises eftersom det saknas kunskap om vad som egentligen behövs (Karlsson, 2002, s 43 ff; Hooks & Farry, 2001, s 4; Sommerville & Sawyer, 1997, s 87). Därför finns det en poäng i att också samla in sådant som kan karaktäriseras som behov eller mål och hantera dem med hjälp av lämpliga attribut.



Figur 3.2.2.1 *Förenklad modell för kravinsamlingsprocessen. Ur Regnell et al. (2003).*

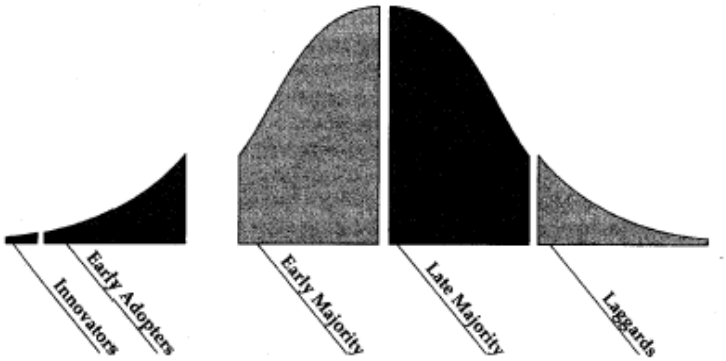
Lite kortfattat handlar analysarbetet om att identifiera, sålla och sätta attribut på behov. Dessa aktiviteter inbegriper var för sig mycket arbete i olika delmoment, och att detta arbete sköts på ett rationellt sätt är avgörande för hur effektiv releaseplaneringsprocessen sedan blir. En stor risk som ett företag möter är att kravmängden blir för omfattande. En viktig del i arbetet är därför att hålla omfånget på databasen hanterligt. Många företag lägger

för lite resurser på att hantera de inkommande kraven[13] och ett minimikrav är att lägga ansvaret för hanteringen av denna kravmängd på särskild personal (Regnell et al., 1998).

Identifiering av behov inbegriper både att ta fram faktiskt existerande behov och att hitta behov som kan skapas. Källorna till behoven finns både att hitta inom den utvecklande organisationen och bland användare och kunder. Tekniska möjligheter kan sporra till att nya behov ”uppfinns” internt i organisationen, men även usergroups och befintliga kundgränssnitt som support-, sälj- och marknadsavdelning är centrala källor för att fånga in krav. Om det redan finns en produkt i produktion är alltså helpdesk, och supportavdelningen med sin erfarenhet av kunders och användares problem en viktig källa till djupare förståelse av de egentliga behoven (Karlsson, 2002, s 47; Alexander & Stevens, 2002, s 50-51). Även utbildare och konsulter för produkten har intensiv kontakt med kunder och användare och därmed god insyn i vad som kan vara besvärligt (Alexander & Stevens, 2002, s 51). En källa som intimt hör samman med marknadsdriven utveckling är inspiration från konkurrenterna och vilka behov de valt att fylla med sitt produktinnehåll.

Clements och Northrop (2001) speciellt pekat på hur man för att göra ett marknadsdrivet företag effektivt måste ha klart för sig vilka gränssnitt mot kunder och användare som finns i största allmänhet. Det är analogt med att utnyttja de gränssnitt som faktiskt finns för att effektivare ta hand om inkommande krav.

En god idé är att regelbundet uppdatera kanalerna för inkommande krav eftersom företaget kommer att lyssna olika mycket på olika källor under produktens livscykel. I alla fall om produkten utvecklar sig enligt Geoffrey Moores synsätt (1999). Enligt Moore genomgår en framgångsrik produkt olika faser där huvuddelen av köparna tillhör olika typgrupper som lyssnar på olika argument vid inköp. Därför går det skiljelinjer mellan faserna. Den största går mellan *Early Adopters* och *Early Majority* och måste överbryggas för att produkten ska överleva.



Figur 3.2.2.1 *The Landscape of the Technology Adoption Life Cycle, ur Moore (1999).*

Innovators: Entusiasterna som är intresserade av all ny teknik. Har ofta inflytande, men har inte lika ofta tillgång till pengarna.

Early Adopters: Visionärerna som har stort inflytande på nya teknologier eftersom de har möjlighet att få fram kapital till satsningar. Godtar produkter som inte är helt färdiga om de får vara med att påverka. De är också

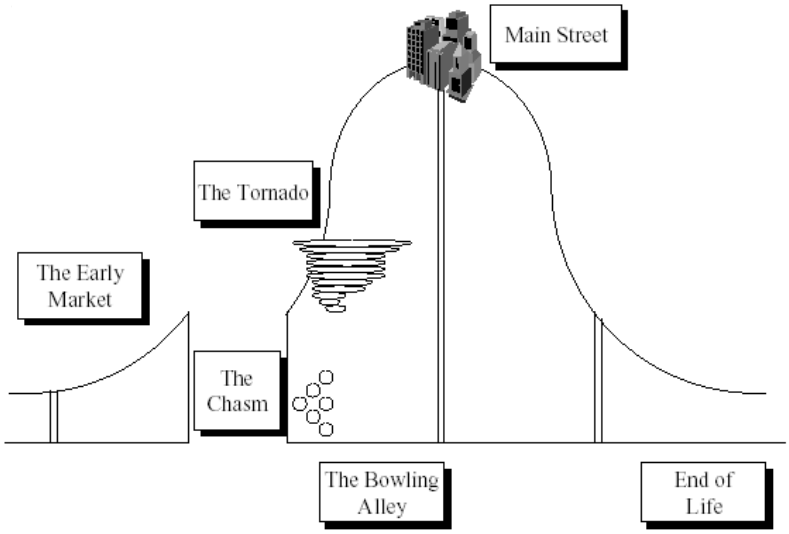
[13] Det har framkommit bland annat under intervjuerna.

intresserade av att få uppmärksamhet riktad till satsningar och drar ofta till sig medial bevakning.

Early Majority: Pragmatiker som inte tror på revolution utan vill ha beprövad teknologi som rekommenderas av pålitliga källor. Köper från marknadsledaren och är de stora inköparna. De är nyckeln till framgång på bred front.

Late Majority: Konservativa med pessimistisk inställning till hur de ska kunna få ut något av teknologiinvesteringar, men är en stor grupp på marknaden. Priskänslig grupp som bara vill betala för det som bevisligen har en nytta.

Laggards: Skeptiker som helst inte vill ha ny teknik (Moore, 1999).



Figur 3.2.2.2 *The Landscape of the Technology Adoption Life Cycle, ur Moore (1998).*

I början kommer många av kraven att uppstå internt då produktens marknad mest handlar om *Innovators* och *Early Majority* (se figur 3.2.2.2). Visionärerna i *Early Majority* kommer också att se nya möjligheter med produkten och komma med specifika krav. Moores recept för att lyckas överbrygga klyftan är att fokusera på ett enda specifikt segment och bli marknadsledande inom det. Källan till kraven är då kundernas specifika behov och det gäller att uppnå en fullständig produkt för att få pragmatikernas förtroende. Den här perioden kallar han *The Bowling Alley* eftersom förhoppningen med att bli marknadsledande inom ett segment är att närliggande segment sedan ska följa efter. *Tornado*-perioden syftar på den kaosartade period när produkten får en bredare spridning. Fokus bör då flyttas till att leverera snabbt och att eliminera konkurrensen för att bli totalt sett marknadsledande. Det blir då viktigare att använda konkurrenternas produktinnehåll som källa till nya krav. När *Main Street* nås är det återigen nöjda kunder som gäller. Tidigare kunde olika samarbeten med andra bolag åstadkomma en helhetslösning i produkten för att tillfredsställa *Early Majority*. Under den här perioden bör dock företaget enligt Moore fokusera på att klara av att leverera en helhetslösning utan partners (Moore, 1995, s 19-26).

Ett problem med behovsidentifieringen vid användning av kunder och användare är att de själva inte alltid är medvetna om sina behov. (Hooks & Farry, 2001, s 4). Maiden och Rugg (1996) har utvecklat ett generellt ramverk för att identifiera behov: Aquisition of

Requirements (ACRE). I deras ramverk ingår både förslag till metoder och också i vilka situationer som de rekommenderas. Exempel på metoder som kan användas är

- Observation av kund och användare i relevant miljö
- Ostrukturerade och strukturerade intervjuer
- Protokollanalys efter att en användare utfört specificerade uppgifter och berättat högt vad han/hon gjort
- Brainstorming
- Rapid prototyping: prototyp utvärderas och kommenteras av kund eller användare
- Scenarioanalys där en kund eller användare får studera scenarier och kommentera
- RAD (Rapid Application Development): en grupp bestående av både utvecklare och kunder arbetar tillsammans för att skapa en gemensam uppfattning om hur det kommande systemet ska se ut.
- Etnografiska metoder där någon från den utvecklande organisationen spenderar längre perioder hos de tilltänkta användarna för att få insikt i deras arbete (Maiden & Rugg, 1996)

Kraven kommer alltså in från en rad olika källor och formuleras av individer med olika bakgrund: supportpersonal, marknadspersonal, utvecklare, säljare osv. Det insamlade materialet kan vara av varierande form och mycket omfattande[14]. Enligt en undersökning bland 36 representanter för marknadsdrivna programvaruutvecklande företag (Regnell et al., 2003) kommer det in upp till 500 krav per vecka, men i genomsnitt ca 14 krav. Av dem är i medeltal 20 % krav av hög kvalitet. Sommerville & Sawyer rekommenderar att det skapas mallar för hur krav samlas in, för att dels bättre lära ut hur krav bör formuleras, och dels för att all relevant information samlas in (1997, s 144). Karlsson rekommenderar bland annat att för varje krav samla in beskrivning, källa, syfte (rationale), nytta, samt vilka kunder och konkurrenter det fokuserar på (Karlsson, 2002, s 53).

Kravdatabasen riskerar att så småningom innehålla dubbletter och krav som blivit inaktuella eftersom ett alternativt krav implementerats istället, samt krav som skjutits upp i tidigare releaser och som aldrig kommer med. Det krävs disciplin och en hel del arbetsinsatser för att hantera materialet[15]. Tidig grovsållning, s k *screening,* av krav innan de läggs in i databasen kan vara en god idé (Regnell et al., 2003). De krav som då bedöms vara ”ointressanta” kastas direkt, givetvis med risken att kasta krav som längre fram visar sig vara värda att implementera. Å andra sidan läggs inte tid på krav som inte är aktuella för implementation och konsekvenserna med en överbelastad kravdatabas som flaskhals är mycket svårare för det fortsatta releaseplaneringsarbetet (Höst et al., 2001). Att inte lägga in dubbletter är också en god ambition, liksom att då krav ändrar status till *Applied* görs en genomgång så att de krav som då blivit inaktuella rensas bort. Tyvärr är detta i nuläget ett ganska manuellt arbete som måste genomföras, men forskning pågår för att utveckla metoder som med automatik söker upp dubbletter bland krav formulerade i naturligt språk. En enkel åtgärd för att inte gradvis öka omfånget på databasen med krav som aldrig väljs ut till implementering är att sätta ett bäst-före-datum på dem[16]. Efter detta datum måste någon typ av beslut fattas om kravet, t ex att ändra tillståndet till *Rejected* eller att sätta ett nytt datum om kravet fortfarande bedöms ha möjlighet att avancera till *Prospect.*

[14] De senaste åren har flera svenska avhandlingar behandlat detta område. Se Natt och Dag, 2002, Karlsson, 1998, Carlshamre, 2001.

[15] Se till exempel Natt och Dag et al. (2002) s 61-84 om hur ett företags kravdatabas blev till en flaskhals.

[16] Används på ett av företagen från intervjudelen.

Vid arkiveringen av nya behov och krav sätts de attribut som är aktuella, vilket minst innebär ett som uttrycker i vilket stadium av sin livscykel det befinner sig i, var kravet kommer från, varför det finns, en beskrivning av kravet och gärna även konkurrent- och kundfokus om det är aktuellt. Därefter måste det göras en grov tidsuppskattning av kravets implementationen för att kunna värdera det. En grovuppskattning av de beroenden som gäller för kravet bör också göras. Dessa två analyser görs lämpligen av en expert, och skickas därför vidare till en sådan. Det innebär att kravet når tillståndet *Assigned*.

3.2.3 Kostnadsuppskattning av krav

Ett sätt att definiera framgångsrika programvaruprojekt är att systemet ska levereras i tid, inom budget och med hög kvalitet (Hughes & Cotterell, 2002, s 79). Det torde gälla för både kontraktsprojekt och kommersiella, även om organisationen själv definierar vad ”i tid” innebär. Hur som helst måste kostnadsuppskattningarna vara rimliga[17] för att kunna planera hur mycket som går att åstadkomma före releasens slutdatum. Kostnaden innebär naturligtvis inte bara lönekostnader, men det är en löpande kostnad som potentiellt kan innebära en risk för företaget. Dåliga tidsuppskattningar kan resultera i både förhöjda kostnader och att releasen inte kan släppas i tid (Höst & Wohlin, 1998, s 332). Det är mycket svårt att göra bra kostnadsuppskattningar. Noggranna uppskattningar riskerar dessutom att bli dyra att genomföra. Därför är det viktigt att lägga ambitionsnivån rätt och lägga mer tid på att uppskatta delar som är mer komplicerade än andra. Det är också onödigt att lägga för mycket tid på tidsuppskattningar av delar som kanske inte kommer att implementeras.

För programvaruutveckling kan tidsuppskattningar specificeras i antal timmar, veckor eller månader för implementation av kravet. Särskilt delar som beräknas ta månader att implementera bör undersökas noga för att minska felmarginalerna. I examensarbetet ligger inte fokus på att detaljstyra *hur* en organisation ska utföra sina aktiviteter utan bara att de *bör utföras*. Därför ges inga detaljerade beskrivningar av t ex hur tidsestimeringar ska göras i modellen, även om det ges en del användbara metoder.

För det första kan estimeringar göras antingen genom en totalbedömning av projektet (*top-down*) där man använder någon typ av algoritm eller parametrar för att beräkna tidsåtgång[18]. Tidsåtgången blir då storleken på system gånger produktiviteten. Om man istället bryter ner projektet i mindre delar som var för sig går att tidsuppskatta på nivån en eller ett par veckor och därefter summera delarna, kallas det *bottom-up*-estimering. Expertbedömningar är vanligt och de använder ofta en kombination av analogier med tidigare projekt och en *bottom-up*-bedömning av tidsåtgången. Om det gäller ändring eller tillägg till existerande produkt görs då först en analys av hur många moduler som är påverkade (Hughes & Cotterell, 2002, s 87).

Tidsuppskattningar kan göras av enstaka experter inom organisationen eller av hela team som i Steen Lichtenbergs successiva princip (Lichtenberg, 2000) för att hantera osäkerheter. Team är också något som Höst & Wohlin pekat på som viktigt för att få bra siffror. De pekar också på fördelarna med att ge estimeringar i intervall, d v s ett lägsta, det mest troliga och ett högsta värde, alternativt ett lägsta och ett högsta med lika sannolikhet att landa någonstans inom intervallet. Dock visar deras resultat att det kan krävas grupper om minst

[17] Sawyer et al, 1999; Carlshamre, 2001; Karlsson, 1998;

[18] Till exempel kan man vikta ihop antalet funktionspunkter med Albrechts funktionspunktsanalys, eller använda Boehms COCOMO-modell.

tjugo experter för att få bra medelvärden (Höst & Wohlin, 1998). Troligen är det rimligast att enstaka experter gör grovuppskattningar, medan uppskattningar i team kan vara användbart inför detaljplaneringen av själva utvecklingsprojekten.

Det finns vissa risker förknippade med den psykologiska effekten av att sätta en tidsuppskattning. Underskattningar av tidsåtgång kan resultera i slarvigt arbete (Hughes & Cotterell, 2002, s 82) och överskattningar kan uttryckas i Parkinsons lag: ”arbetet kommer att expandera för att fylla tillgänglig tid” (Parkinson, 1957). Ett annat motto säger att mer personal i ett projekt kommer att sluka mer tid för koordinering och kommunikation, och Brookes lag säger att tillsättning av mer personal till ett redan försenat projekt orsakar ännu mer förseningar (Brooks, 1995). Oavsett vilken metod som används är det viktigt att kostnadsuppskattningarna är rimliga för att inte orsaka tidsförskjutningar och skenande kostnader.

3.2.4 Prioritering och värdeanalys av krav

Höst et al. har som tidigare nämnts, visat att tidig bortprioritering av krav med uppenbart låg prioritet kan minska risken för att kravdatabasen blir en flaskhals (Höst et al., 2001). Prioritering utpekas redan av Lubars et al. (1992, s 6) som en viktig aktivitet för marknadsdriven utveckling. De anmärker dock att det då inte fanns några explicita metoder för hur prioriteringen skulle gå till. Eftersom releasedatum ibland är viktigare att hålla än att produkten innehåller precis alla planerade features, spelar prioriteringen stor roll för att inrikta arbetet på rätt område. Hur prioriteringen skall gå till finns det fortfarande ingen konsensus om, men Karlsson & Ryan har visat att parvisa jämförelser mellan krav på ganska hög abstraktionsnivå kan vara användbart och göra processen snabbare (Karlsson, 1998). Metoden har sina fördelar eftersom resultatet lätt kan visualiseras och beskrivs noggrannare nedan.

Vems prioritering det handlar om måste klargöras. För att göra en så användbar prioritering som möjligt krävs goda kunskaper om den avsedda marknaden. Viktiga kundgrupper måste få en produkt som sammantaget är begärlig och inte bara en kompromiss av alla intressenters prioriteringar. I examensarbetet kommer två typer av prioriteringsarbeten att beröras. Dels handlar det om den prioritering som görs då innehållet i produkten ska väljas ut. Denna prioritering kallar vi hädanefter *värdeanalys*[19]. Dels handlar det om den prioritering som är aktuell under implementeringsfasen. Den görs av en programvaruarkitekt för att planera i vilken ordning som funktionalitet ska implementeras.

Eftersom resurserna för att framställa produkten är begränsade och produktens featureinnehåll många gånger har en avgörande inverkan på om kunden kommer att köpa den, bör urvalet av features ske på ett någorlunda informerat sätt. Vid värdeanalys är kostnad och värde centrala begrepp[20], men särskilt när det gäller vilket ”värde” som avses finns det skillnader. Begreppet kan orsaka en del förvirring eftersom det inbegriper metoder där värdet mäts i kronor och ören, men det kan också handla om metoder där resultatet blir en rankning av olika produktfeatures. Lehmann och Winer (2002, s 308) klassificerar metoderna i fem grupper:

[19] Inom forskningen används också begreppet Cost-Value analysis för denna typ av prioritering. Även kostnader för implementeringen måste räknas in.
[20] Se t ex Carlshamre, 2001, s 187-215, Karlsson, 1998, s 115-129.

1. Industriella ingenjörsmetoder där kunder uppskattar hur produktförändringar skulle förändra värdet på produkterna.
2. Övergripande uppskattning av kundvärde, d v s hur kunder värderar värdet av produkten i pengar (willingness-to-pay).
3. Samma som 2. men på produktattributnivå. Görs som benchmarking[21] eller jämförelser mellan olika attribut.
4. Direkta frågor till kunder om värde på olika produktattribut (ingen jämförelse).
5. **Kunder får ranka hur viktiga olika produktattribut är. Inbegriper också jämförelser med konkurrenternas produktattribut.**

När det gäller produktutveckling är det svårt att sätta ett numeriskt värde på hur mycket en ny feature skulle kunna vara värd i pengar. Den typen av metoder är mer aktuell när det handlar om prissättningsfrågor på den redan utvecklade produkten. Däremot är den femte kategorin intressant med kundprioriteringar av attribut både från den egna produkten och från konkurrenternas. Den typen av rankningar ingår i aktuell forskning som ibland benämns Cost-Value Analysis och ibland bara prioritering. Karlsson et al. (Karlsson, 1997) menar att relativa resultat har visat sig mer pålitliga och snabbare att åstadkomma än absoluta bedömningar. Därför har parvis jämförelse, t ex mellan olika konkurrerande features, fördelar som prioriteringsmetod[22].

Parvisa jämförelser kan åstadkommas med en rad olika metoder, som utvärderats av Karlsson et al. (1997). Den metod som i praktiken visat sig användbar är en variant på Analytic Hierarchy Process (AHP), vilket egentligen är en metod för beslutsfattande.

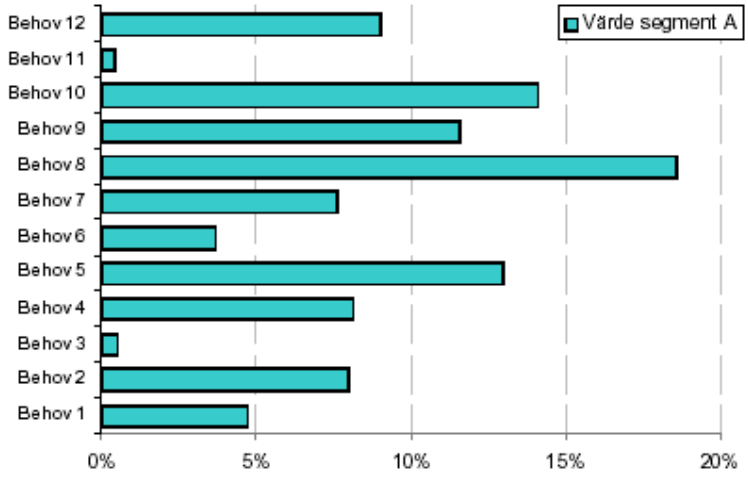
Vi antar att vi vill göra en prioritering av en uppsättning features. För det första bör kraven vara på samma hierarkiska nivå. Ofta är krav arrangerade i en hierarkisk struktur där ett krav fungerar som en slags ”rubrik” som sedan specificeras i underkrav. Väl valda kundrepresentanter ska sedan göra jämförelserna. De kan representera olika segment och vara ”riktiga” kunder, eller representeras internt via exempelvis produktledare eller marknadsförare. Dock ska man vara medveten om att representanter hämtade från utvecklings- eller marknadsföringssidan tenderar att ha en mer positiv attityd till betydelsen av features jämfört med supportpersonal (Faulk et al., 2000). Alla krav på samma hierarkiska nivå jämförs med varandra, vilket sammanlagt blir n(n-1)/2 jämförelser. Varje jämförelse ges sedan ett värde enligt skalan:

1 Kraven är lika viktiga
3 Måttlig skillnad i hur viktiga kraven är
5 Essentiell skillnad i hur viktiga kraven är
7 Stor skillnad i hur viktiga kraven är
9 Extrem skillnad i hur viktiga kraven är

Resultatet presenteras sedan i ett liggande stapeldiagram där längden på varje stapel representerar varje features relativa prioritet. Det blir alltså möjligt att se hur mycket viktigare en feature är jämfört med andra. Det finns också en inbyggd felkontroll eftersom användaren kan bestämma att feature A är viktigare än B, och B är viktigare än C och sedan ändå bestämma att C är viktigare än A. En nackdel med AHP är att den skalar dåligt, vilket ställer krav på ett lämpligt urval att göra jämförelserna på. Eventuellt kan kraven delas in i

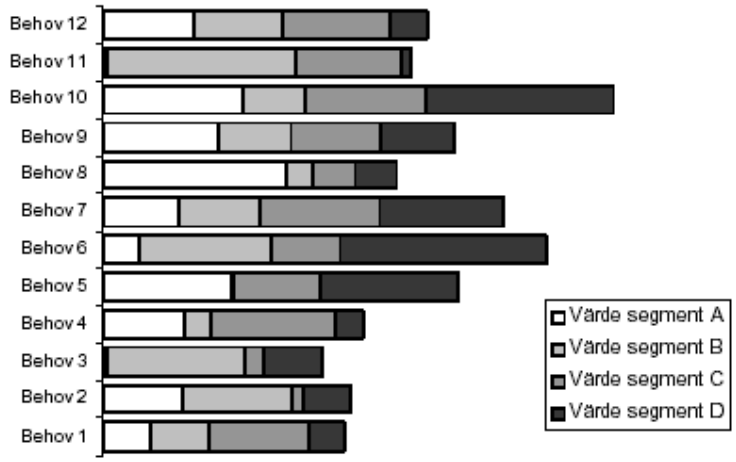
[21] Benchmarking är att jämföra sitt företag med ett annat som anses vara bra inom det man vill förbättra sig. Det kan gälla processer, produkter eller servicenivå.
[22] En del företag använder sig även internt av absoluta värden, se Carlshamre, 2001, s 187-215. Nackdelen är att många krav riskerar få samma prioritering.

grupper, inom vilka AHP appliceras (Karlsson, 1997). Karlsson et al., rekommenderar också att jämförelserna görs tillsammans av en grupp, vilket skulle stimulera till diskussion av kraven. Resultatet av en värdeanalys, genomförd med hjälp av programmet Focal Point, inom ett kundsegment skulle kunna se ut som i figur 3.2.4.1.



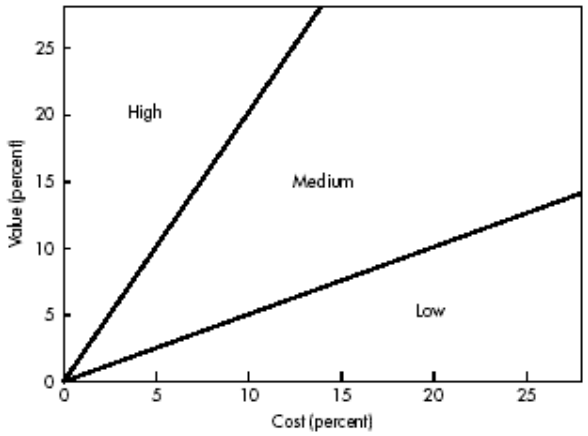
Figur 3.2.4.1 *Värdediagram för 12 behov i kundsegmentet A.*

Kundvärdeanalys i flera marknadssegment kan också slås samman till ett gemensamt värdediagram, se figur 3.2.4.2. Här blir information om hur viktigt ett visst segment är osynlig. Poängen är att genom att göra kundvärdeanalys för prioriterade segment också kunna göra välavvägda beslut om hur nöjda ett visst segment de skulle bli om en särskild uppsättning av krav valdes ut.



Figur 3.2.4.3 *Värdediagram för fyra marknadssegment utan hänsyn till hur viktigt respektive segment är.*

Informationen om kundvärden ställs med fördel mot information om implementationskostnad. Informationen kan sedan föras in i ett diagram som i figur 3.2.4.3 och användas som diskussionsunderlag vid releaseplaneringen. ”High” innebär att kvoten mellan kundvärde och kostnad är hög och därför en attraktiv grupp att välja features ur. Denna typ av prioritering kan utföras med eller utan stödjande programvara, men resultatet av de forskningsmaterial inom detta område som hänvisas till i examensarbetet har resulterat i kommersiell programvara för beslutsstöd (och kravhantering) vid releaseplanering.



Figur 3.2.4.3 *Resultatet av värdeanalysen enligt AHP kan plottas i ett diagram mot kostnaden för implementeringen (bild hämtad ur Karlsson, 1998, s 119).*

En stor fördel med denna typ av prioritering är att resultatet lätt går att visualisera. Motsvarande värdeanalyser bör också göras för att uppskatta värdet av features internt för företaget. Hur bra en viss feature går att marknadsföra är en viktig del, liksom om implementeringen av en viss feature tekniskt sett ger fördelar som går att dra nytta av vid vidareutvecklingen av produkten (”teknisk höjd”)[23]. Tyvärr tar inte jämförelserna hänsyn till beroenden, utan analysresultatet måste användas på ett medvetet sätt.

3.2.5 Beroendeanalys

En komplicerande faktor är de beroenden som kan finnas mellan krav. Till exempel kan ett högprioriterat krav kräva att ett lågprioriterat krav redan finns implementerat (Carlshamre, 2002). En industriell undersökning av Carlshamre et al. visar att endast ett fåtal krav är helt singulära och att ca 20 % av kraven står för ca 80 % av alla beroenden i kravmängden. Det finns också, enligt författarna, en tendens till att beroenden är mer värderelaterade i marknadsdriven utveckling och mer funktionalitetsinriktad i kontraktsdriven. Värderelaterade beroenden handlar om att ett krav kan påverka kundvärdet av ett annat krav, t ex kan tillgång till en detaljerad online-manual påverka värdet av en pappersmanual. Funktionella beroenden syftar mer på en relation mellan krav som att ett krav kräver att ett annat finns, t ex e-mail kräver nätverk, en printer kräver en drivrutin. Ibland kan det också vara nödvändigt att de implementeras i viss ordning (add-funktioner implementeras före delete-funktioner). Ytterligare ett exempel på funktionella beroenden är att ett krav gör att ett annat blir inaktuellt, t ex så kan bilder i ett dokument skapas antingen via ett separat ritprogram eller med intern ritmodul i applikationen (Carlshamre et al., 2001).

För att skaffa sig en mer nyanserad bild av de beroenden som kan finnas bland föreslagna features som ska ingå i en release, kan den kategorisering av beroenden som Carlshamre gjort vara användbar (Carlshamre, 2002).

[23] Begreppet ”teknisk höjd” har framkommit under intervjuer.

AND	A printer requires a driver to function, and the driver requires a printer to function.
REQUIRES	Sending an e-mail requires a network connection, but not the opposite.
TEMPORAL	The function *Add object* should be implemented before *Delete object*.
CVALUE	A detailed on-line manual may decrease the customer value of a printed manual.
ICOST	A requirement stating that "no response time should be longer than 1 second" will typically increase the cost of implementing many other requirements.

Tabell 3.2.5.1 *Kategorisering av olika typer av beroenden. Hämtad ur Carlshamre, "A Usability Perspective on Requirements Engineering", s 171-172, 2001.*

Om alla typer av beroenden mellan alla krav skulle spåras och dokumenteras skulle det resultera i allt för många kombinationer. Det skulle bli svårt att hålla dokumentationen av alla beroenden uppdaterade. Varje förändring av ett krav skulle resultera i att ett större eller mindre antal beroenden skulle förändras. Den nivån är alltså inte rimlig för beroendeanalys. Vad som behövs är ett stöd för att genomföra releaseplaneringen, och då är det lämpligt att fokus ligger på beroenden mellan högnivåkrav. Förslagsvis är temporala relationer mindre intressanta vid releaseplanering, men användbara vid iterationsplanering.

3.2.6 Releaseplanering

När processen kommit så långt som till releaseplanering bör det finnas ett bra underlag för att fatta beslut. Releaseplanering är inget arbete som genomförs en gång per release. Beroende på omvärldsfaktorer, som konjunkturläge, konkurrens, mm. kan revideringar krävas även mitt under utvecklingen av en release. Att ha tillgång till ett gott underlag gör det rimligare att fatta ett bra första beslut om releaseinnehållet och därefter kunna löpande uppdatera på ett medvetet sätt.

Vad som slutligen hamnar i releasen bör förhoppningsvis vara baserat på information om slutdatum, kostnad för varje krav, värde för kund och beroenden. Framför allt är det en balansering mellan värde för kund och kostnad för implementering, men kompliceras av olika typer av beroenden. Det är alltså relativt komplext att planera en release men tyvärr inget som går att undvika. Det underlättar arbetet väsentligt om underlaget för beslutet är bra, men exakt hur det ska gå till finns det ingen allmän uppfattning om, vilket Carlshamre noterat (2002). Överhuvudtaget finns det inte mycket praktisk hjälp att hitta i litteraturen, och risken finns att stödverktyg blir förenklade. Också Karlsson som utvecklat metoden för releaseplanering grundat på parvisa jämförelser, intar en ödmjuk inställning till vad metoder och modeller för releaseplanering kan åstadkomma.

Ytterligare en komplicerande faktor har med konkurrenternas agerande att göra. En marknadsdriven organisation måste löpande vara uppdaterad med konkurrenternas produkter och hur dessas features positioneras i aktuella marknadssegment (se kapitel 3.4). Det innebär att kravmängden kan komma att genomgå större förändringar under projektets gång och då gäller det att ha bra bakgrundsinformation för att något smidigare kunna fatta bra beslut.

3.3 Marknadsdriven kontra kontraktsdriven utveckling

En heltäckande teori för vad som är utmärkande för marknadsdriven programvaruutveckling saknas. Dock kan karaktäristika hämtade ur observationer från industrin vara användbara. Nedan finns en sammanställning av de egenskaper som skiljer de två utvecklingsinriktningarna åt. Tabellen är inte på något sätt föreskrivande och naturligtvis finns det företag som har element från både kontraktsbaserad- och marknadsdriven utveckling.

Tabell 3.3.1 är ursprungligen sammanställd av Carlshamre (Carlshamre, 2001) som har hämtat informationen från Kamsties et al. (1998); Keil & Carmel (1995); Lubars et al. (1993); Novorita & Grube (1996); Potts (1995) och Yeh (1992). Den har sedan modifierats av Natt och Dag (Natt och Dag, 2002) som använt ytterligare observationer från Potts (1995) och Lubars et al. (1993) och byggt på med insikter från med Robertson & Robertson (1999). Slutligen kompletteras tabellen med andra aspekter hämtade från Karlsson (2002); Novorita et al. (1996); Sawyer el al. (1999); Lewis & Rieman (1994), Lehmann & Winer (2002) och Lauesen (2002).

Karakteristik	**Kontraktbaserad utveckling**	**Marknadsdriven utveckling**
Primärt mål	Uppfylla kravspecifikationen.	Time to market. Kraven hellre stryks eller planeras för nästa release istället för att tillåta försening av produktreleasen. Kostnadsreducering.
Framgångsmått	Tillfredställelse, acceptans.	Försäljning, marknadsandelar, produktrecensioner, lönsamhet.
Livscykel	En release, sedan underhåll.	Flera releaser, så länge det finns en marknad för produkten.
Kravens ursprung	Insamling, analys, validering.	Påhittad. En ny feature tillåts av antingen marknaden (marknads-avdelningen) eller teknologin.
Kravspecifikation	Används som kontrakt mellan kund och leverantör.	Existerar sällan eller finns informellt. Kraven diskuteras verbalt. Läses ej av kunden.
Användare	Känd eller lättidentifierbar.	Svår att identifiera eller okänd till en början.
Kund	Programvarubeställare. Kontraktsförhandlare. En eller fåtal.	Representant för olika marknader. Huvudkunden kan få anpassad programvara. Många.
Fysiskt avstånd till användarna	Vanligtvis litet.	Vanligtvis stort.
Huvudintressent	Kundorganisationen.	Utvecklingsorganisationen.
Speciella kravhanteringsaktiviteter	Insamling, modellering, validering, konfliktlösning.	Hantering av konstant ström av inkommande krav. Balansering av innehåll och kostnad med hjälp av prioritering. Releaseplanering.
Utvecklarnas anknytning till programvaran	Kortfristig (till projektets slut).	Långfristig, kan t ex medverka i underhåll av programvara.
Validering	Fortlöpande process.	Väldigt sent, t ex vid mässor.
Kravhanteringsstandarder och explicita metoder.	Vanligt.	Sällsynt.
Iterativa tekniker	Mindre vanligt.	Mer vanligt.
Domänexperter tillgängliga bland utvecklarna	Mer vanligt.	Mindre vanligt (produktutvecklingen bryter ofta ny mark).

Systemmodellering	Baserad på kundens omgivning.	Begreppsmässiga modeller av kundens verksamhetsmiljö är hypotetiska.
Konkurrenter	Inga.	Många.
Val av innehåll, leveranstid och prissättning	Specificerad i kontrakt.	Stor frihet.

Tabell 3.3.1 *Kontraktsbaserad- mot marknadsdriven programvaruutveckling (från Carlshamre, 2001; Natt och Dag, 2002; Karlsson, 2002; Novorita et al. 1996; Sawyer el al., 1999; Lewis & Rieman, 1994, Lehmann & Winer, 2002).*

Primärt mål: Det huvudsakliga målet med marknadsdriven programvaruutveckling är vad som kallas *time to market*, d v s att få ut produkten på marknaden så snabbt som möjligt (Novorita & Grube, 1996). När datumet för releasen närmar sig är det viktigare att hålla sig till tidsplaneringen än att ha med alla features. Därför händer det ofta att krav som bidrar minst till produktens framgång på marknaden stryks eller planeras för nästa release istället (Lubars et al., 1993). Det är dock inte alltid som den högsta prioriteten är att hinna först, utan att hitta sitt s.k. marknadsfönster, d v s när det är mest gynnsamt att släppa produkten. Hur som helst är det avgörande *när* produkten är färdig. I kontraktsbaserad programvaruutveckling är målet att uppfylla de krav som finns i kravspecifikationen. När och hur produkten ska levereras framgår av kontraktet.

Framgångsmått: Framgång med marknadsdriven utveckling ses mer från ett ekonomiskt perspektiv medan det i kontraktsbaserad utveckling handlar om hur väl kundens krav uppfylls. När en produkt släpps på en marknad är det viktigt att den får bra recensioner så att företaget kan vinna marknadsandelar och sälja sin produkt. För att lyckas kommersiellt räcker det inte bara med att ha den bästa produkten, utan företaget måste ha goda kunskaper om sina marknader. Vad den centrala kunskapen handlar om, förklaras senare i avsnittet om marknadsteori (kap 3.4).

I kontraktsbaserad utveckling är framgångsmåtten baserade på acceptans av och tillfredställelse hos kunden. Eftersom de ekonomiska detaljerna bestäms innan projektets start och specificeras i kontraktet är de inte lika relevanta för produktens framgång.

Livscykel och utvecklarnas anknytning till programvaran: Programvarans livscykel är i ett kontraktsbaserat projekt kortfristig jämfört med ett lyckat marknadsdrivet projekt. I det förstnämnda är det huvudsakliga målet att få fram en färdig produkt som kunden kan använda. Det kan sedan förekomma underhåll av programvaran, till exempel för att korrigera fel eller om ändrade arbetsmetoder hos kunden måste återspeglas i programvaran.

Sawyers et al. (1999) menar att det är troligt att det görs mer ansträngningar i kravhanteringsprocessen för att göra produkten lätt att underhålla om tanken är att släppa en produkt på marknaden med regelbundna releaser. Trots att det är viktigt att släppa en bra första release, måste programvaran kunna anpassas för att möta ändrade marknadsförhållanden, nya behov och att utnyttja tekniska möjligheter (Sawyer et al., 1999).

Kravens ursprung och kravspecifikation: En stor skillnad mellan kontraktsbaserad- och marknadsdriven utveckling är kravens ursprung och specifikation. Eftersom det inte finns en särskild beställare så blir insamlingsprocessen av kraven annorlunda i marknadsdriven utveckling jämfört med kontraktsbaserad utveckling. Kravspecifikationen, vars huvudsyfte är att användas som kontrakt mellan kund och leverantör, blir mindre formell, om den ens existerar (Sawyer et al., 1999; Lubars et al., 1993). I marknadsdriven utveckling kan kraven för en helt ny produkt inte erhållas externt utan får istället formuleras internt i organisationen (Lubars et al., 1993). Detta kan göras av antingen marknadsavdelningen eller

utvecklingsavdelningen, beroende på om det är marknaden som tillåter en ny feature eller om det är teknologin.

Användare, kund, fysiskt avstånd: I tabellen ses kund som den som köper produkten och användare som slutanvändare av produkten. Inom kontraktsbaserade projekt är kunden den som beställer produkten och den som är kontraktsförhandlare. I sådana projekt är användaren känd eller lättidentifierbar eftersom de ofta är anställda på beställarorganisationen. I marknadsdrivna projekt däremot är kunden ibland slutanvändaren och ibland en organisation som rymmer många olika användargrupper. Även om utvecklingsorganisationens huvudmål är att lyckas på marknaden, kan de ha företag som huvudkunder och som kan få specialanpassade versioner av programvaran som finns tillgänglig på marknaden. Det kan även vara så att det är storkundens krav som styr vad som ska ingå i produkten. Det är också betydligt svårare att identifiera användarna eftersom marknaden kan bestå av tusentals människor med olika bakgrund.

Oavsett om produkten utvecklas för en marknad eller en specifik kund så har studier visat att det inte alltid är så lätt för utvecklarna att föreställa sig vem slutanvändaren är. Problemet finns även i kontraktsbaserade projekt där utvecklingsorganisationen ofta är en underleverantör som automatiskt hamnar ett steg längre bort från slutanvändaren (Lubars et al., 1993).

Huvudintressent: Leffingwell & Widrig definierar en intressent som ”någon som kan påverkas materiellt genom implementationen av det nya systemet eller applikationen” (Leffingwell & Widrig, 2000, sid 40). Huvudintressent skulle då kunna beskrivas som den som materiellt sätt påverkas mest av implementationen och vars behov och intressen därför tas störst hänsyn till.

I kontraktsbaserad utveckling är det naturligtvis kundorganisationen som är huvudintressent. I marknadsdriven utveckling läggs stor vikt på att ta reda på vad marknaden vill ha och vad som kan säljas, men trots det är det inte kunderna och användarna som är huvudintressenter. Materiellt sett är det företaget som utvecklar produkten som påverkas mest av produkten via intäkterna som genereras, och därför är huvudintressent.

Speciella kravhanteringsaktiviteter: Ett typiskt kontraktsbaserat projekt genomgår faserna kravinsamling, analys av kraven samt validering. Insamlingen av kraven utgår från kundens organisation där olika intressentgrupper får ställa krav på vad produkten ska klara av. Därefter görs analys av kraven till exempel som användningsfall och aktörer och en kartläggning av hur användningsfallen är besläktade. Kunden får sedan validera kraven och avgöra att rätt produkt kommer att produceras. Beställaren är delaktig i processerna och godkänner via valideringen. På så sätt kan konflikter förebyggas eftersom det är kunden själv som fäller avgörandet att produkten är på rätt väg.

En marknadsdriven organisation har inte sin kravinsamling fullt så begränsad till början av projektet, om det inte handlar om en helt ny produkt. En organisation som haft en produkt på marknaden ett tag kommer att löpande få in nya krav via bland annat support- och säljavdelningen. Dessa måste tas om hand inför planeringen av nya releaser. Vid val av innehåll till en ny release kommer aldrig alla features att hinna implementeras om den ska bli färdig i rimlig tid och då krävs det kostnadsuppskattningar för kraven. Releaseplaneringen handlar sedan om en balansering av prioriteringar och kostnader.

Validering: Validering och verifiering är två begrepp som ofta dyker upp tillsammans. De har inte samma betydelse, men de blandas lätt ihop (Sommerville, 2001, s 420). Sommerville refererar till Boehm (1979) som beskriver skillnaden mellan validering och verifiering:

> ”Validation: Are we building the right product?”
> “Verification: Are we building the product right?”

Verifiering innebär alltså att programvaran testas för att se om den följer specifikationen, medan validering handlar om att möta kundens förväntningar. Vilken tidpunkt i utvecklingen som det sistnämnda utförs vid skiljer sig åt mellan kontraktsbaserad- och marknadsdriven utveckling. I kontraktsbaserad utveckling utförs acceptanstester fortlöpande, där kunden ger klartecken om utvecklingen av programvaran är på rätt väg. I marknadsdriven utveckling är det inte lika lätt att få klartecken från kunderna. Valideringen kan inte utföras förrän stora delar av implementeringen är slutförd och prototyper eller färdiga produkter kan presenteras på t ex mässor (Sawyer et al., 1999). Tidiga prototyper kan visserligen valideras av potentiella kunder eller fokusgrupper, men den egentliga valideringen görs genom marknadens emottagande.

Kravhanteringsstandarder och explicita metoder: Användning av processtöd för marknadsdrivna utvecklingsprocesser är mindre moget än hos kontraktsdrivna utvecklingsföretag och har till och med setts som sekundärt i förhållande till produkten (Carmel & Sawyer, 1998). Förändringar i kravmängden för att anpassa produkten till marknadens svängningar kräver större arbetsinsatser om särskilda standarder och metoder måste följas. Samtidigt är skälen att använda strukturerade modeller som i kontraktsdriven utveckling många och ett antal artiklar pekar på behovet av explicita metoder för att åstadkomma stabila produkter (Novorita & Grube, 1996; Carmel & Sawyer, 1998, s 11).

Iterativa tekniker: Iterativa metoder är att föredra framför vattenfallsmetoden eftersom riskerna i programvaruutveckling lättare kan upptäckas och hanteras då (se kap 3.5). Trots det visar studier att iterativa metoder inte är lika vanliga i kontraktsbaserad utveckling som i marknadsdriven utveckling (Lubars et al., 1993). Vad det beror på finns det ingen förklaring till.

Domänexperter tillgängliga bland utvecklarna: I kontraktsdrivna projekt finns det ofta domänexperter tillgängliga bland utvecklarna. De har minst lika bra förståelse för domänen som kunden har, ibland till och med bättre. I marknadsdrivna projekt är det inte alltid någon som har en helhetsbild av problemområdet. Detta kan bero på att marknadsdrivna utvecklingsorganisationer satsar på att bryta ny mark. Istället brukar sådana projekt ta råd från konsulter. (Lubars et al., 1993)

Systemmodellering: Systemmodellering handlar om hur systemet ska fungera i kundorganisationens verksamhetsmiljö samt hur kompatibelt det ska vara med andra programvaror och tekniska standarder. Programvaran ska kanske kunna installeras på ett speciellt operativsystem och använda sig av specifika protokoll. Detta kan göras fullt ut när en kund finns närvarande och utvecklingsorganisationen kan se vad kunden har tillgång till, men i marknadsdriven utveckling är modellerna endast hypotetiska (Sawyer et al., 1999). Eftersom kundunderlaget då ofta är blandat byggs mer flexibilitet in i produkten (Potts, 1995)

Konkurrenter: Ett företag måste alltid ha koll på sina konkurrenter, oavsett om de har en kontraktsbaserad- eller marknadsdriven utveckling. Skillnaden är att det i kontraktsbaserade projekt finns eventuell konkurrens fram tills att ett kontrakt skrivs, medan konkurrenterna i

marknadsdrivna projekt alltid är närvarande. I tabellen ovan avses konkurrens efter att utvecklingen har börjat. Då har antingen ett kontrakt skrivits under och utvecklingsprocessen kan genomföras utan hänsyn till konkurrens, eller så utvecklas produkten för en marknad där det ofta finns många konkurrenter.

Val av innehåll, leveranstid och prissättning: När det gäller innehåll, leveranstid och prissättning för en produkt är valfriheten större i marknadsdrivna projekt än i kontraktsbaserade. I det sistnämnda kommer utvecklingsorganisationen överens med kunden om sådana detaljer innan projektets start, som sedan specificeras i kontraktet. I marknadsdrivna projekt är utvecklingen till en viss del styrd av omvärlden, men trots det så har utvecklingsorganisationen stor frihet i sina val. Det gäller dock att företaget har bra koll på omvärlden och utvecklar en produkt som de kan konkurrera med och framförallt sälja.

3.4 Marknadsteori

Varje produkt är resultatet av marknadsmässiga överväganden. I förslaget till en förändrad RUP finns delar som kommer att påverka produktinnehållet. De marknadsteoretiska begrepp som berörs beskrivs här översiktligt. Exakta arbetsmetoder kommer inte att beröras.

3.4.1 Marknadsstrategier

För att ett företag framgångsrikt ska kunna lansera sina produkter och konkurrera på marknaden måste det ha en strategi att gå efter. Ansoffs matris är ett verktyg som ofta används när företag ska bestämma sina produkt- och marknadsstrategier. Strategierna används för att inrikta verksamheten och följer nedan:

	Nuvarande Produkter	Nya Produkter
Nuvarande Marknader	Marknadspenetrering	Produktutveckling
Nya Marknader	Marknadsutveckling	Diversifiering

Tabell 3.4.1 *Marknadsstrategier (Ansoffs matris)*

1. **Marknadspenetrering**: vinna fler marknadsandelar med sina nuvarande produkter i nuvarande marknader.
2. **Marknadsutveckling:** hitta eller skapa nya marknader för sina nuvarande produkter, d v s utöka kundbasen.
3. **Produktutveckling:** utveckla nya produkter som kan vara intressanta för nuvarande marknader, d v s fördjupa den existerande kundbasen.
4. **Diversifiering:** utveckla nya produkter för nya marknader, d v s expandera företagets affärsidé (Kotler, 2003, s 100; Karlsson, 2002, s 16).

Alla marknadsstrategier byggs på med segmentering, målmarknadsföring (*market targeting*) och positionering. Ett företag upptäcker behov och grupper på marknaden, riktar in sig på de grupper vars behov det kan uppfylla och positionerar sig hos målmarknaden för att bli erkänd för sina erbjudanden och sin image. Dessa aktiviteter beskrivs nedan.

3.4.2 Marknadssegmentering

Ett företag kan inte tillfredställa alla kunders behov, dels för att det inte ofta är möjligt att fråga alla kunder vilka behov de har och dels för att företaget inte kan be alla sina kunder att prioritera sina behov. En produkt som släpps på en marknad kan ju ha flera tusen kunder. Dessutom kan olika kunder ha olika intressen och om produkten skulle utvecklas för att täcka delar av alla kunders behov, skulle den sammanlagda produkten inte bli tillräckligt bra för någon kund. Därför måste marknaden segmenteras, d v s kunderna delas

in i grupper efter behov. Då kan företaget försöka tillfredställa varje segments behov istället för varje enskild kunds behov. Vidare är det viktigaste inte att tillfredställa alla segmentens alla behov, utan att tillgodose rätt behov hos prioriterade segment.

Kundmarknaden kan t ex delas in efter följande segmenteringsvariabler:

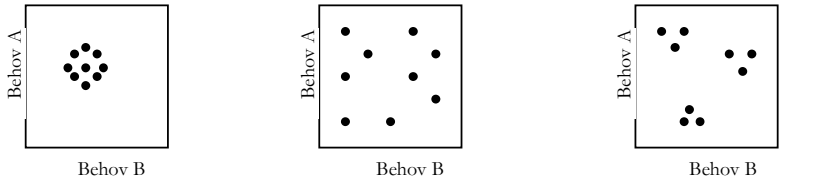
Geografisk:	Region	Sverige, USA, Norden, Skåne,...
	Stadsstorlek	10 000-250 000, över 1 000 000,...
	Täthet	Tätort, landsbygd,...
	Väderstreck	Nordlig, sydlig...
Demografisk:	Ålder	Under 6, 20-35, 65+, ...
	Familjestorlek	1-2, 3-4, 5+,...
	Civilstånd	Gift, sambo,...
	Kön	Man, kvinna
	Inkomst	Under 100 000, 100 000-200 000,...
	Yrke	Student, läkare, pensionär,...
	Utbildning	Gymnasium, högskola,...
	Religion	Kristen, muslim, hindu,...
	Ras	Asiat, afrikan,...
	Generation	Baby boomers, Generation X,...
	Nationalitet	Tysk, svensk, fransk...
	Social klass	Arbetarklass, medelklass, ...
Psykografisk:	Livsstil	Kulturorienterad, sportorienterad,...
	Personlighet	Ambitiös, auktoritär, impulsiv, ...
Beteende:	Tillfällen	Regelbundet, speciella tillfällen,...
	Värdering	Kvalitet, service, fart,...
	Användarstatus	Potentiell användare, flitig användare,...
	Användargrad	Liten, medel, stor,...
	Lojalitetsstatus	Ingen, medium, stark, absolut,...
	Intresse för produkt (readiness stage)	Omedveten, medveten, informerad, intresserad,...
	Attityd mot produkten	Entusiastisk, positiv, neutral, negativ,...

Tabell 3.4.2 *Övergripande segmenteringsvariabler för kundmarknaden. Ur Kotler, 2003, s 288 och Karlsson, 2002, s 18.*

Alla segmenteringsvariabler är inte alltid lika lämpliga. Det spelar t ex troligen ingen roll vilket kön kunden har när han eller hon köper bröd. För att segmenteringen ska vara användbar måste segmenten vara:

1. **Tillräckligt omfattande** – I termer av potentiell försäljning ska segmentet vara tillräckligt stort för att företaget överhuvudtaget ska satsa på det. Antalet kunder i segmentet är mindre viktigt, så länge det är lönsamt.
2. **Identifierbara** – Segmenten ska vara identifierbara så att de kan kallas för annat än ”segment A” och ”segment B”. T ex ”de bilintresserade”. Segmentidentiteten underlättar även de strategiska och taktiska besluten.
3. **Tillgängliga** – Det är viktigt att kunna nå ut till kunderna i segmenten för att kunna marknadsföra och sälja produkten. Bilintresserade kunder kan t ex nås genom tidningar, medan kunder som t ex gillar blå färg kan vara lite svårare att nå.
4. **Skiljaktiga** – De olika segmenten ska helst reagera på olika sätt för minst några beståndsdelar av företagets erbjudande. Om gifta och ogifta män gillar samma bilmärke tillhör de inte olika segment .
5. **Enhetliga** – Segmentet ska kunna antas vara homogent och en genomsnittsmedlem ska vara rimligt lik alla andra medlemmar i segmentet.
6. **Stabila** – Eftersom framtiden baseras på data från det förflutna bör segmenten vara stabila över tiden (Lehmann & Winer, 2002, s 164; Kotler, 2003, s 286).

Ett sätt att hitta segmentmönster är att identifiera preferenssegment. Detta görs genom att undersöka hur kunderna värderar olika produktegenskaper. På så sätt hittas likheter och skillnader mellan segmenten. Tre typer av mönster kan uppstå:

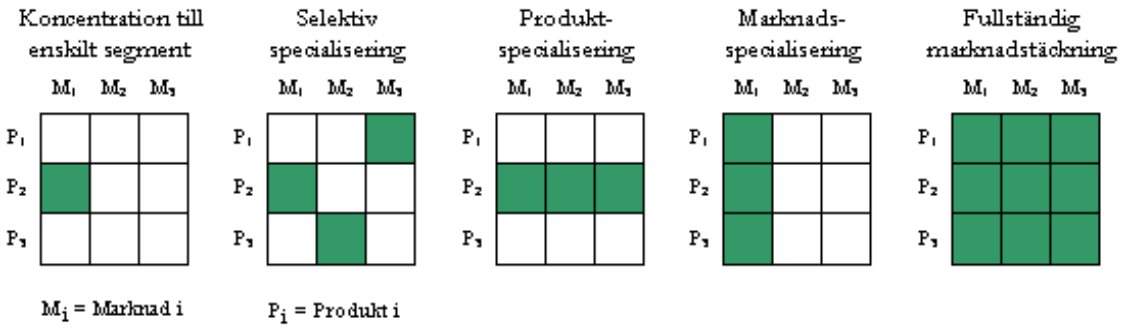


Figur 3.4.1 *a) Homogena preferenser* *b) Diffusa preferenser* *c) Klustrade preferenser*

- **Homogena preferenser** – alla kunderna har ungefär samma preferenser. Marknaden behöver därför inte segmenteras. Alla liknande varumärken skulle konkurrera om samma kunder.
- **Diffusa preferenser** – kunderna har väldigt olika preferenser. Första varumärket på marknaden placeras förmodligen i centrum för att tilltala flest människor. Efterföljande varumärken kan antingen placeras i närheten av den första för att konkurrera om marknadsandelar, eller längre ut mot sidorna för att tilltala de som inte blev tillfredställda av tidigare varumärken.
- **Klustrade preferenser** – marknaden visar tydliga preferenskluster. Dessa brukar kallas för naturliga marknadssegment. Första företaget på marknaden har tre valmöjligheter. Det kan antingen positionera sig i mitten för att locka alla grupper, positionera sig i det största marknadssegmentet eller utveckla flera varumärken för att positionera sig i alla segmenten (Kotler, 2003, s 283; Karlsson, 2002, s 17).

3.4.3 Målmarknadsföring (*market targeting*)

Efter att marknaden segmenterats måste företaget värdera attraktionskraften hos varje segment och bestämma vilka segment det ska rikta sig mot. Detta görs med hänsyn till företagets mål och tillgångar. Hur attraktivt ett segment är avgörs med hjälp av karaktäristika som storlek, tillväxt, lönsamhet etc. När värderingen är gjord kan företaget välja mellan fem olika mönster för val av marknadssegment (*target market selection*).



Figur 3.4.2 *Olika mönster för val av marknadssegment*

- **Koncentration till enskilt segment** – produkten släpps endast på en specifik marknad, t ex Porsche koncentrerar sig endast på sportbilsmarknaden. På detta sätt får företaget bättre kunskap om segmentets behov och kan därför positionera sig bättre inom segmentet. Detta är ett riskfyllt val eftersom segmentets beteende kan

ändras och påverka företagets vinst eller så konkurrens kan dyka upp i segmentet och ta marknadsandelar från företaget. Å andra sidan menar Moore (1998) att det är genom att bli marknadsledande inom ett segment (*The Bowling Alley)* som framgång på bredare front nås genom att sedan erövra angränsande segment.

- **Selektiv specialisering** – attraktiva segment som var för sig garanterar intäkt väljs ut, trots att de kanske inte ens har någon samverkan. Ett exempel är medieföretag som äger radiostationer. Olika stationer kan rikta sig till olika åldersgrupper. På detta sätt fördelas riskerna.
- **Produktspecialisering** – samma produkt säljs till flera segment, t ex mikroskop som säljs till universitet, kommersiella laboratorier och staten. Företaget får ett bra rykte inom produktområdet, men nackdelen är att produkten kan bli ersatt av en helt ny teknologi efter ett tag.
- **Marknadsspecialisering** – alla behov inom samma segment täcks med hjälp av olika produkter. Exempel på detta är företag som endast tillverkar produkter till laboratorier. Fördelen med detta är att företaget blir känd inom segmentet och kan lättare sälja ytterligare produkter inom segmentet. Nackdelen är att segmentet kan få finansiella problem som påverkar företagets intäkter.
- **Fullständig marknadstäckning** – företaget täcker alla behov inom alla segment med hjälp av olika produkter. General Motors är ett exempel på ett sådant företag (Kotler et al., 2002, s 340 – 346; Kotler, 2003, s 299).

Val av mönster är beroende av företagets tillgångar. Ett företag med begränsade tillgångar skulle välja någon av strategierna där koncentrationen ligger på en eller några fåtal marknadssegment, medan större företag kan satsa på olika marknadssegment och erbjuda speciella produkter till varje segment.

3.4.4 Positionering

Positionering är den image som byggs upp för ett företag eller dess produkter. Detta görs bl a med hjälp av slogans som kortfattat ska ge kunderna en bra anledning till varför de ska köpa just det företagets produkter. Ett exempel är Volvo som har positionerat sig som den säkraste personbilen. Kunder som sätter säkerhet före allt annat väljer oftast Volvo vid val av bil. Om bilen sedan tekniskt sätt verkligen är den säkraste eller inte behöver inte spela någon roll. Positionering handlar alltså om vilken position företaget eller produkten har i kundens medvetande (Kotler, 2003, s 308).

Hur positioneringen går till beror på om företaget har den ledande produkten på marknaden eller inte. Om företaget är marknadsledande är målet med positioneringen att försvara marknadsandelar genom att visa hur bra produkten är. Annars handlar det oftast om att kontra marknadsledaren eftersom kunderna alltid jämför nya produkter med den som är ledande på marknaden (Karlsson, 2002, s 23). Några exempel är:

- ”Coke is it.” – “Pepsi. A choice of a new generation”
- “Kodak. Show your true colors.” – “Fuji. Always a better film.”

Ibland händer det att positioneringen blir så lyckad att varumärket sammankopplas med produkten. Exempel på sådana varumärken är Xerox, Kleenex och Rollerblades (Karlsson, 2002, s 24). Detta är en klar fördel om företaget endast har en produkt som de konkurrerar med. Men om företaget vill utöka sin produktportfölj kan det uppstå problem med att sälja

de nya produkterna. Xerox var t ex länge synonymt med kopiering och företaget skulle få svårt att sälja produkter som inte har med kopiering att göra.

Geoffrey Moore (1999, s 144-147) koncentrerar sig mer på tekniska produkter och menar att nyckeln till framgång inte är att tänka på positionering som ett sätt att lättare sälja en produkt, utan som ett sätt att lättare kunna köpa den. Med det tankesättet kan företagen koncentrera sig på vad kunderna verkligen vill ha och sedan låta försäljningen av produkten sköta sig själv. Om produkten är lätt att köpa så blir den automatiskt även lätt att sälja. Hur detta ska göras beror på vilka köparna är. Produktlivscykeln har olika faser där köparna tillhör olika typgrupper och med olika argument för att köpa (se kap 3.2.2 Behovsanalys). Moore föreslår fyra fundamentala steg för positionering hos de olika typgrupperna:

1. **Namnge rätt** för entusiasterna. Potentiella kunder köper inte produkter vars namn de inte kan uttala. Detta är den minimala ansträngning för positionering som krävs för att produkten lätt ska kunna köpas av entusiasterna som är intresserad av ny teknik.
2. **För vem och varför** för att nå visionärerna. Kunderna måste veta vem som ska använda produkten och varför innan de köper den. Detta är den minimala extra ansträngning för positionering som krävs för att produkten lätt ska kunna köpas av visionärerna.
3. **Konkurrens och differentiering** är viktigt för pragmatikerna. Kunderna vill kunna jämföra produkten med andra liknande produkter för att veta vad de ska förvänta sig och hur mycket de ska betala. Detta är den minimala extra ansträngning för positionering som krävs för att produkten lätt ska kunna köpas av pragmatiker.
4. **Finansiering och framtid** prioriteras av konservativa. Kunderna känner sig inte säkra vid köp av produkten om de inte vet om försäljaren är tillräcklig stark för att finnas kvar på marknaden och fortsätta investera i produktkategorin. Detta är den minimala extra ansträngning för positionering som krävs för att produkten lätt ska kunna köpas av de konservativa.

Dessa fyra steg ska utföras vid lämpliga tidpunkter under produktens utveckling på marknaden så att typgrupperna är redo för produkten.

3.4.5 Konkurrentanalys

Att hitta konkurrenterna kan vara svårare än vad man kan tro. Ofta har företagen fler konkurrenter än vad de räknat med. Coca-cola konkurrerar t ex med Pepsi om cola-sugna kunder, men även med Ramlösa om läsksugna kunder och Marabou om sötsugna kunder. Konkurrenterna är alltså de företag som tävlar om potentiella kunders pengar med antingen samma typ av produkt (direkta konkurrenter), liknande produkter, eller substitut (båda indirekta konkurrenter) (Karlsson, 2002, s 25).

På senare år har företagen även börjat konkurrera med hjälp av nya teknologier (Kotler, 2003). Bokhandlarna och skivförsäljarna har fått nya konkurrenter som amazon.com, som bedriver försäljning av böcker, musik mm via Internet. Eftersom det är så bekvämt för kunderna att i lugn och ro sitta framför datorn och beställa varor har det blivit allt vanligare att företagen använder sig av Internet för att nå ut till kunderna.

Efter att konkurrenterna har identifierats kan företaget försöka ta reda på vilka mål och strategier de har på de olika marknadssegmenten. Detta kan göras genom undersökning av

tidigare och nuvarande marknadsföringsstrategier. Vissa konkurrenter kanske försöker vinna marknadsandelar genom att pressa priserna medan andra håller priserna höga och istället har hög kvalitet på service eller lockar med andra tjänster. Företaget kan även försöka ta reda på hur konkurrenterna planerar att utöka sina verksamheter, även om det är en svår uppgift att genomföra på laglig väg. Det kan också vara bra att göra undersökningar för att kunna förutspå hur konkurrenterna reagerar vid t ex prisjusteringar eller olika marknadsföringskampanjer. (Kotler, 2003, s 248-251; Lehmann & Winer 2002, s 117-126)

Att ta reda på konkurrenternas styrkor och svagheter är viktigt för att veta när de utgör hot och när det egna företaget kan hota konkurrenterna. Enligt konsultföretaget Arthur D. Little kan företag ha en av sex olika marknadspositioner (Kotler, 2003, s 248-249; Karlsson, 2002, s 26):

1. **Dominerande**. Företaget kontrollerar beteendet hos andra konkurrenter och har många strategiska möjligheter.
2. **Stark**. Företaget kan ta oberoende initiativ utan att äventyra sin långsiktliga position oberoende av konkurrenternas agerande.
3. **Fördelaktig**. Företaget har en potentiell styrka och en god möjlighet att förbättra sin position.
4. **Tveksam**. Företaget gör tillräckligt för att kunna fortsätta sin verksamhet, men är beroende av det dominerande företaget och har få möjligheter att förbättra sin position.
5. **Svag**. Företaget har en otillfredsställande prestanda med dock möjlighet för förbättring vid förändring.
6. **Oduglig**. Företaget har otillfredsställande prestanda och saknar möjlighet till förbättring.

Dessutom bör varje företag kontrollera tre faktorer vid analys av konkurrenter (Kotler, 2003, s 250; Karlsson, 2002, s 27):

1. **Share of market**. Konkurrentens andel av målmarknaden.
2. **Share of mind**. Andelen av kunder som skulle nämna konkurrenten som svar på frågan "Vilket är det första företaget som du tänker på inom detta område?".
3. **Share of heart**. Andelen av kunder som skulle nämna konkurrenten som svar på frågan "Vilket företag skulle du helst vilja köpa produkten av?".

Det företag som ökar mest på de två sistnämnda faktorerna kommer oundvikligen även öka sina marknadsandelar och sin vinst.

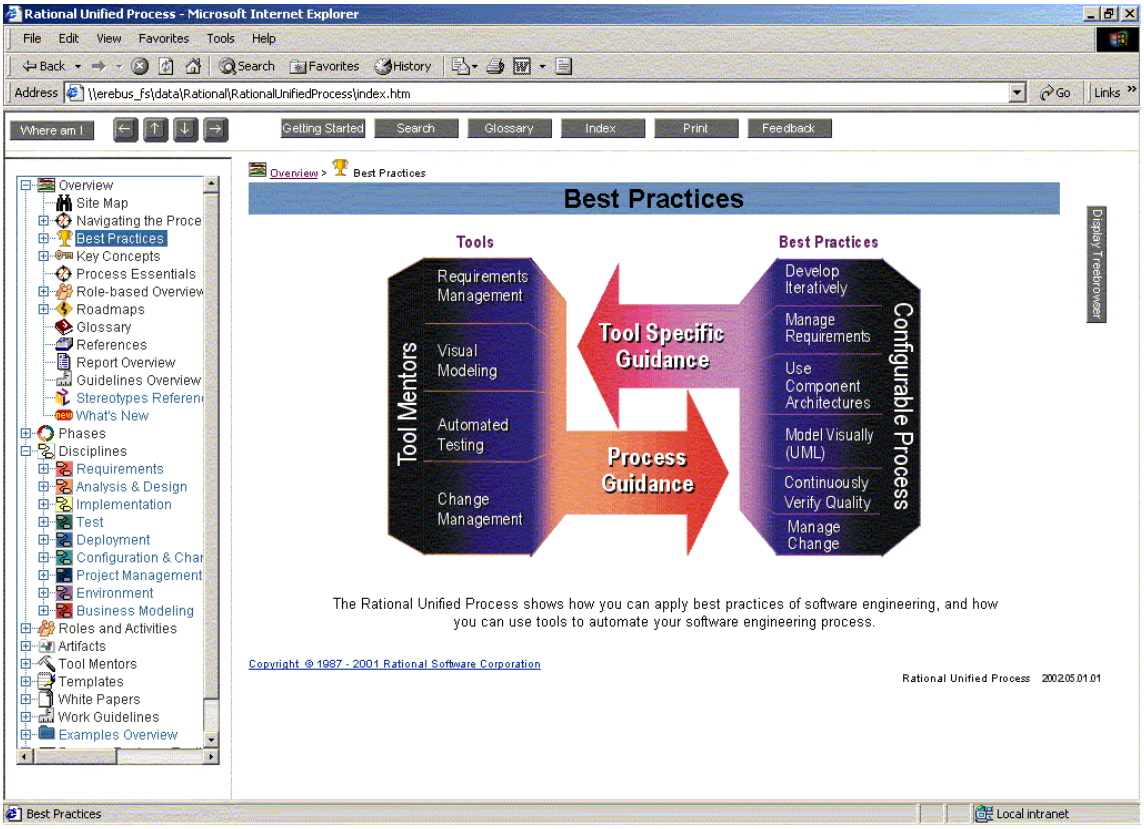
När konkurrentanalysen är klar är det dags att välja konkurrensstrategi. Vilken strategi företaget ska välja beror på vilken roll de spelar på marknaden: ledare, utmanare, imiterare eller nischföretag (*market nicher*).

- Marknadsledaren har den största marknadsandelen för produkten och för att fortsätta dominera vill ledaren öka efterfrågan på marknaden, skydda sina marknadsandelar och kanske även försöka öka sina marknadsandelar.
- Utmanaren attackerar ledaren och andra konkurrenter för att ta marknadsandelar.
- Imitatören gör inget, utan följer bara strömmen och nöjer sig med sina marknadsandelar.
- Nischföretaget koncentrerar sig på de små marknadssegmenten vars behov inte har blivit tillfredsställda av de större företagen (Kotler, 2003).

3.5 *Rational Unified Process*

Bildmaterialet är hämtat från Rational Unified Process Version 2002.05.00.

Rational Unified Process (RUP) är en process som används som ramverk inom programvaruutveckling (*software engineering*) och som består av riktlinjer för hur en utvecklingsorganisation ska gå till väga för att producera högkvalitativ programvara.
Den är anpassningsbar och sägs kunna modifieras efter varje enskild organisations behov. Dessutom levereras RUP som ett webbgränssnitt och är därmed lättillgänglig för utvecklare.

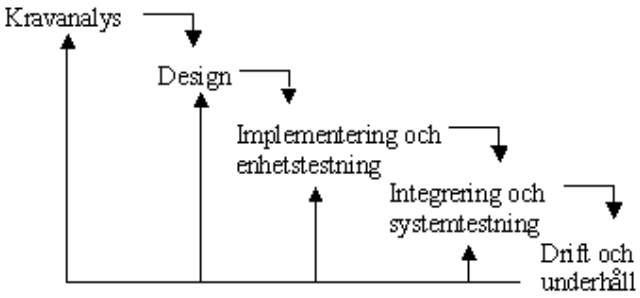


Figur 3.5.1 *Rational Unified Process*

Grundorsaker till problem vid programvaruutveckling kan vara dålig förståelse för slutanvändarnas behov, bristande förmåga att sköta kravändringar eller att programvaran inte håller en tillräcklig hög kvalitet. För att lösa dessa problem finns det ”väl beprövade metoder”, som sägs användas av många framgångsrika företag:

1. Utveckla programvara iterativt.
2. Hantera krav.
3. Använd komponentbaserade arkitektur.
4. Modellera programvara visuellt.
5. Verifiera programvarans kvalitet kontinuerligt.
6. Hantera ändringar av programvaran (Kruchten, 2002, s 5-6).

RUP är en iterativ process som bygger på vattenfallsmodellen och spiralmodellen. Vattenfallsmodellen består av olika faser som avverkas en efter en.



Figur 3.5.2 *Vattenfallsmodellen. Ur Sommerville (2001)*

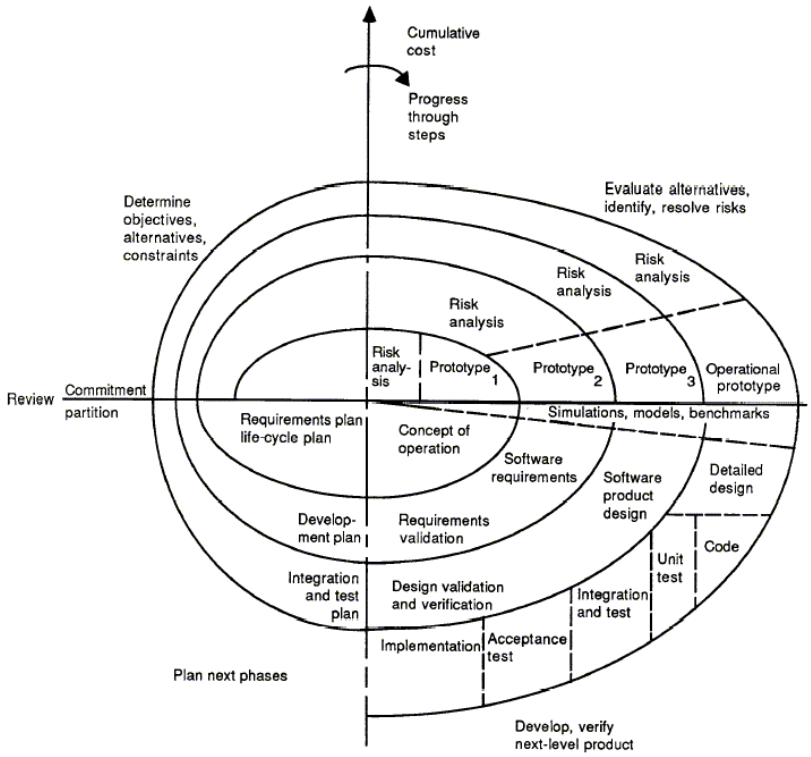
I första fasen definieras och analyseras krav, vilket resulterar i systemspecifikationen. Under nästa fas, designas hela systemet baserat på kraven som bestämdes i föregående fas. Därefter implementeras och testas delsystem under den tredje fasen. Fjärde fasen går ut på att sätta ihop hela systemet och testa det för att sedan överlämna det till kunden. Den femte och slutliga fasen är till för drift och underhåll av programvaran (Sommerville, 2001, s 45).

Nackdelen med den här modellen är att risker skjuts fram i tiden, vilket kan bli kostsam för utvecklingsföretaget eftersom det innebär att en eller flera faser måste upprepas för att korrigera eventuella fel. Om t ex utvecklarna vid systemtestning upptäcker att ett krav har misstolkats kan de bli tvungna att göra om stora delar av designen och skriva om programkoden som sedan ska fungera med hela systemet igen. Det kan resultera i förseningar och i värsta fall nedläggning av projektet. Tabell 3.5.1 är hämtad från Davis (1993, s. 25) och visar den relativa reparationskostnaden i olika stadier av programvaruutvecklingen.

Fas	Relativ kostnad för reparation
Krav	0.1-0.2
Design	0.5
Kodning	1
Enhetstest	2
Acceptanstest	5
Underhåll	20

Tabell 3.5.1 *Kostnad för att reparera programvara i relation till livscykelstadium*

En alternativ utvecklingsmodell är spiralmodellen. Spiralmodellen är en riskdriven utvecklingsmodell där varje fas i utvecklingsprocessen representeras av ett varv i spiralen. Riskerna uppskattas och reduceras genom utveckling av prototyper som utvärderas. Till en början (fas 0) bestäms det om utvecklingen är genomförbar. Om så är fallet börjar första varvet (fas 1) där konceptidén fastställs. Efterföljande faser avser kravdefiniering, design, integration och test. Varje fas är indelad i fyra delar: målsättningar (för fasen), riskuppskattning och reducering, utveckling och validering samt planering inför nästa fas (Boehm, 1988).



Figur 3.5.3 *Spiralmodellen. Ur Boehm (1988)*

RUP, som är en iterativ process, är en blandning av vattenfallsmodellen och spiralmodellen. Från spiralmodellen kommer den medvetna riskhanteringen och att ta in feedback från kunder och användare, gärna via prototyper. Fasindelningen liknar vattenfallsprocessen fast endast med fyra faser. Varje fas avlutas med en milstolpe och beslut om projektet ska gå vidare eller ej. Faserna är följande:

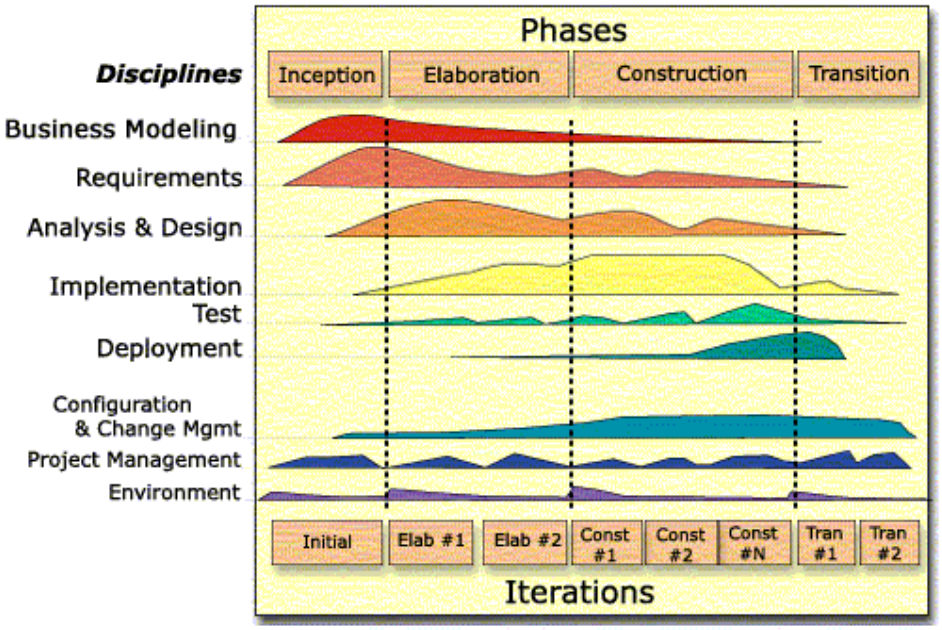
- Förberedelsefasen (*Inception*) – förstå vad projektet går ut på, utföra en analys av problemet, specificera visionen och affärsnyttan med produkten samt definiera projektets omfattning. Fokus ligger på de övergripande kraven samt projektets avgränsningar. Fasen avslutas med milstolpen Livscykelsmål.

- Etableringsfasen (*Elaboration*) – planera projektaktiviteter och resursfodringar, specificera och förfina systemkraven samt utarbeta en första genomförbar arkitektur. Den mest kritiska fasen där fokus ligger på kraven, arkitekturen samt processen. Fasen avslutas med milstolpen Livscykelsarkitektur.

- Konstruktionsfasen (*Construction*) – visionen, systemarkitekturen samt planerna för produkten färdigställs samtidigt som produkten byggs upp. Fokus ligger på implementeringen. Fasen avslutas med milstolpen Initial funktionsduglighet.

- Överlämningsfasen (*Transition*) – betatestning utförs för att sedan kunna överlämna produkten till användarna. Även tillverkning, leverans, utbildning, support och underhåll av produkten ingår tills alla är tillfredställda. Fokus ligger på driftsättning. Fasen och eventuellt, om alla är nöjda, hela livscykeln avslutas med milstolpen Produktutgåva (Leffingwell & Widrig, 2000; Kruchten, 2002).

Varje fas utförs i en eller flera iterationer där syftet med varje iteration, liksom i spiralmodellen, är att mildra aktuella risker i utvecklingen. (Leffingwell & Widrig, 2000)



Figur 3.5.4 *Processtruktur*

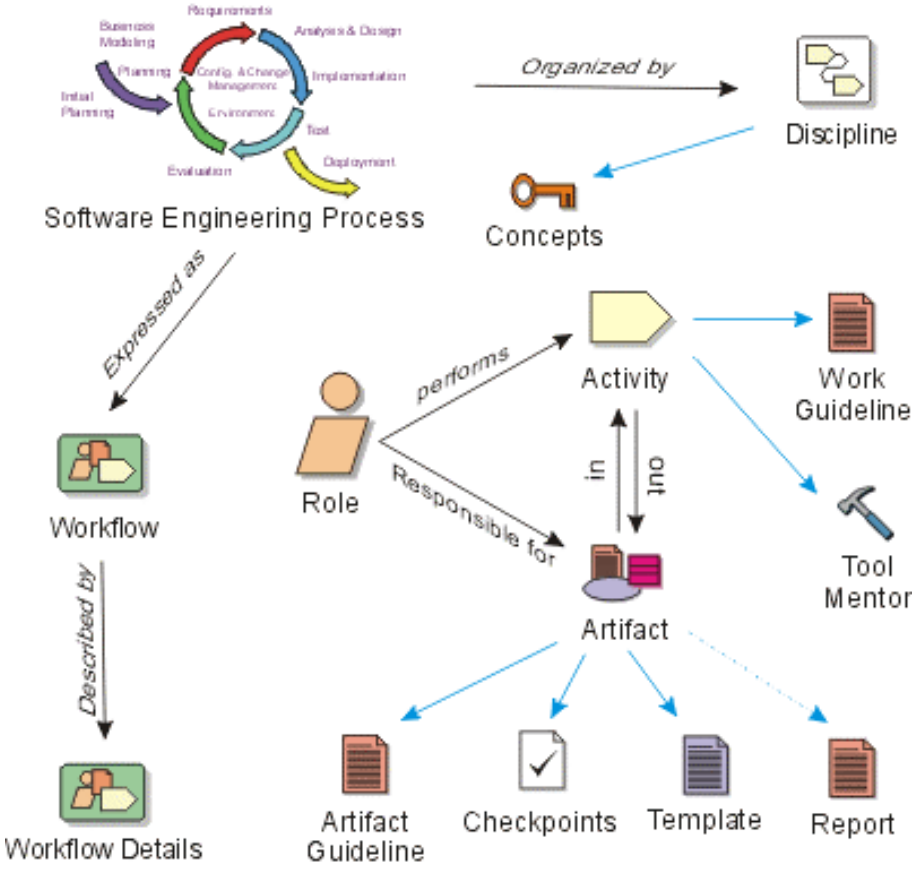
Figur 3.5.4 visar processtrukturen i två dimensioner. Den horisontella axeln representerar tiden och visar vilka faser och antal iterationer som ingår i processens livscykel. Den vertikala axeln visar arbetsbördan för respektive disciplin (beskrivs nedan).

Figur 3.5.5 *Iterativ process*

RUP består av fyra grundläggande processelement:

1. Discipliner
2. Roller
3. Aktiviteter
4. Artefakter

Utöver roller, aktiviteter och artefakter finns det andra processelement som är till för att göra processen mer lättförståelig och lättanvänd. Dessa övriga element är riktlinjer (*Guidelines)*, malldokument (*Templates*), verktygsguider (*Tool Mentors*) samt begrepp (*Concepts*). Hur och när dessa ska användas beskrivs i *arbetsflödesdetaljerna* som beskriver arbetsflödet (se nedan).



Figur 3.5.6 *Grundläggande begrepp i RUP*

3.5.1 Discipliner

Discipliner beskrivs i arbetsflöden (*Workflow*) som säger *när* en uppgift ska utföras inom ett visst arbetsområde. Varje arbetsflöde visar det sekventiella arbetet med hjälp av arbetsflödesdetaljer (*Workflow Detail*). Det finns totalt nio discipliner i RUP varav sex är för utveckling och tre är stödjande. Disciplinerna för utveckling är följande:

- *Business Modeling* – förstå målorganisationens struktur och identifiera förbättringar för dess problem.
- *Requirements* – fånga, organisera och hantera krav effektivt samt ge grunden för kostnad- och tidsberäkningar för att utveckla systemet.
- *Analysis & Design* – förstå kraven och med hjälp av kravspecifikationen utforma systemdesignen.
- *Implementation* – dela in systemet i mindre subsystem och sedan implementera och testa varje enhet för sig.

- *Test* – verifiera att subsystemen samverkar felfritt och är integrerade i systemet utan några problem. Hitta fel och säkerställa att kravspecifikationen har följts.
- *Deployment* – paketering, distribution, installation samt support för programvaran. Även testning av systemet i den slutgiltiga miljön ingår.

De stödjande disciplinerna är följande:

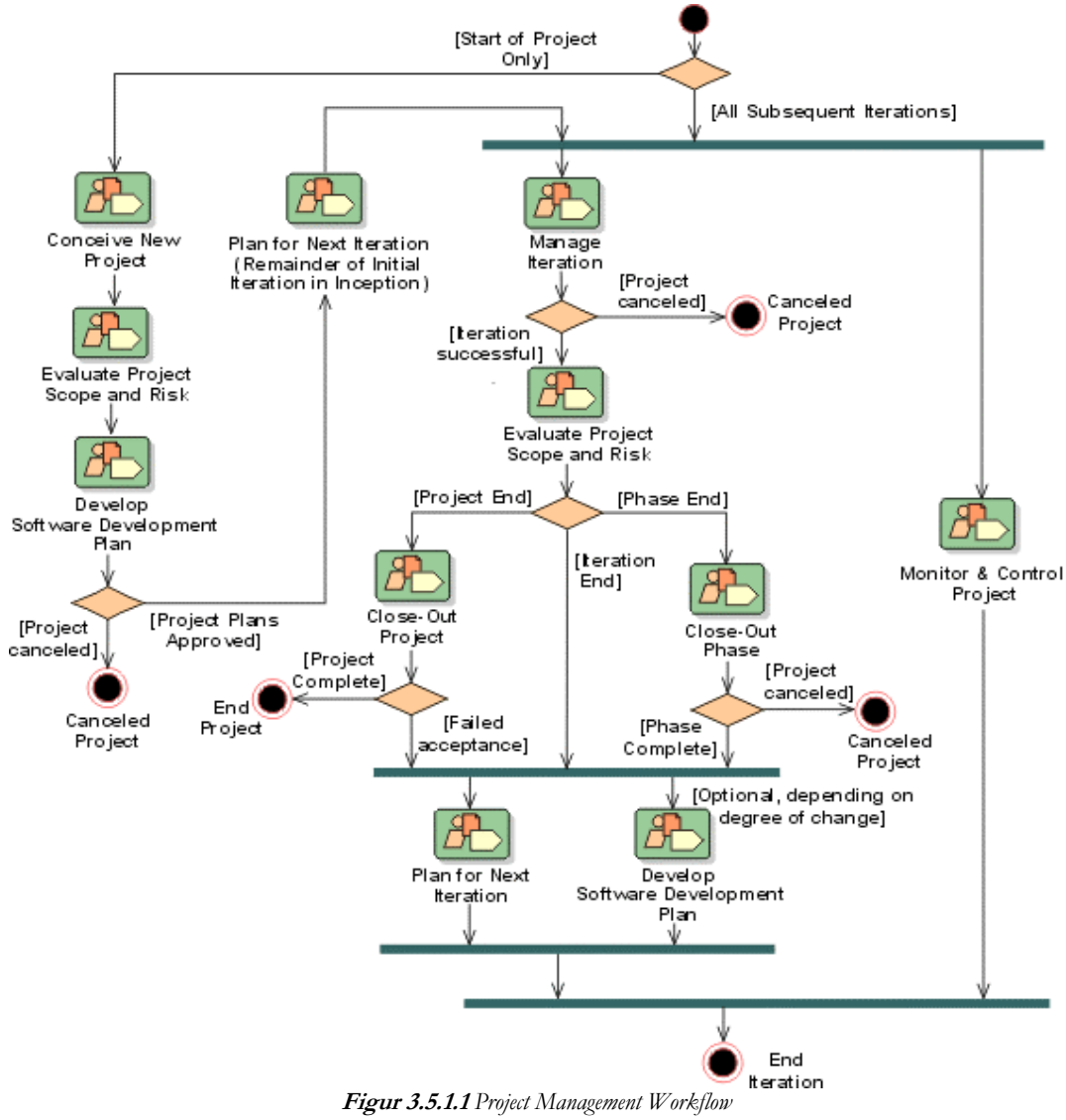
- *Configuration & Change Management* – följa upp och bevara integriteten hos projektets dokument när de förändras.
- *Project Management* – planera faser och iterationer, riskhantering samt mätvärden och övervaka det iterativa projektet.
- *Environment* – hjälpa utvecklingsorganisationen med metoder, processer och verktyg.

De discipliner som kommer att analyseras i examensarbetet är *Project Management*, *Business Modeling* och *Requirements*. Bildmaterialet från RUP som beskriver dessa tre arbetsflöden för respektive disciplin visas nedan tillsammans med kortfattade beskrivningar. Vi väljer dock att inte översätta namnen på arbetsflödetsdetaljerna och aktiviteterna till svenska eftersom de är tagna direkt ur RUP.

Project Management Workflow
Syftet med arbetsflödet är att ge

- ett ramverk för hantering av programvaruintensiva projekt
- praktisk vägledning för planering, bemanning, utförande och övervakning av projekt
- ett ramverk för att hantera risker (Kruchten, 2002, s113-14)

Arbetsflödet karaktäriseras som stödjande disciplin och är tänkt att fungera som ramverket där projekt skapas och drivs. Alla andra discipliner används för att genomföra projektet. Det är främst stöd för iterationsplanering, riskhantering, mätningar och godkännande av olika projektmoment som arbetsflödet är tänkt att erbjuda. Det finns en tydlig gränsdragning för vad som *inte* täcks in av arbetsflödet: personalfrågor, budgetarbete och hantering av kontrakt med leverantörer och kunder. Ansvarig för samtliga aktiviteter utom granskningarna (utförs av *Project Reviewer*) är projektledaren (*Project Manager*). Projektledaren ska allokera resurser, sätta prioriteringar och koordinera samarbetet med kunder och användare.



Figur 3.5.1.1 *Project Management Workflow*

Arbetsflödesdetalj	Aktivitet
Conceive New Project	Identify and Assess Risks Develop Business Case Initiate Project Project Approval Review
Evaluate Project Scope and Risk	Identify and Assess Risks Develop Business Case
Develop Software Development Plan	Develop Measurement Plan Develop Risk Management Plan Develop Product Acceptance Plan Develop Problem Resolution Plan Develop Quality Assurance Plan Define Project Organization and Staffing Define Monitoring and Control Processes Plan Phases and Iteration Compile Software Development Plan Project Planning Review

Plan for Next Iteration	Develop Iteration Plan Develop Business Case Iteration Plan Review
Manage Iteration	Acquire Staff Initiate Iteration Assess Iteration Iteration Evaluation Criteria Review Iteration Acceptance Review
Close-Out Phase	Prepare for Phase Close-Out Lifecycle Milestone Review
Close-Out Project	Prepare for Project Close-Out Project Acceptance Review
Monitor & Control Project	Monitor Project Status Report Status Schedule and Assign Work Handle Exception and Problems PRA Project Review

Tabell 3.5.1.1 *Aktiviteterna inom respektive arbetsflödesdetalj i Project Management Workflow.*

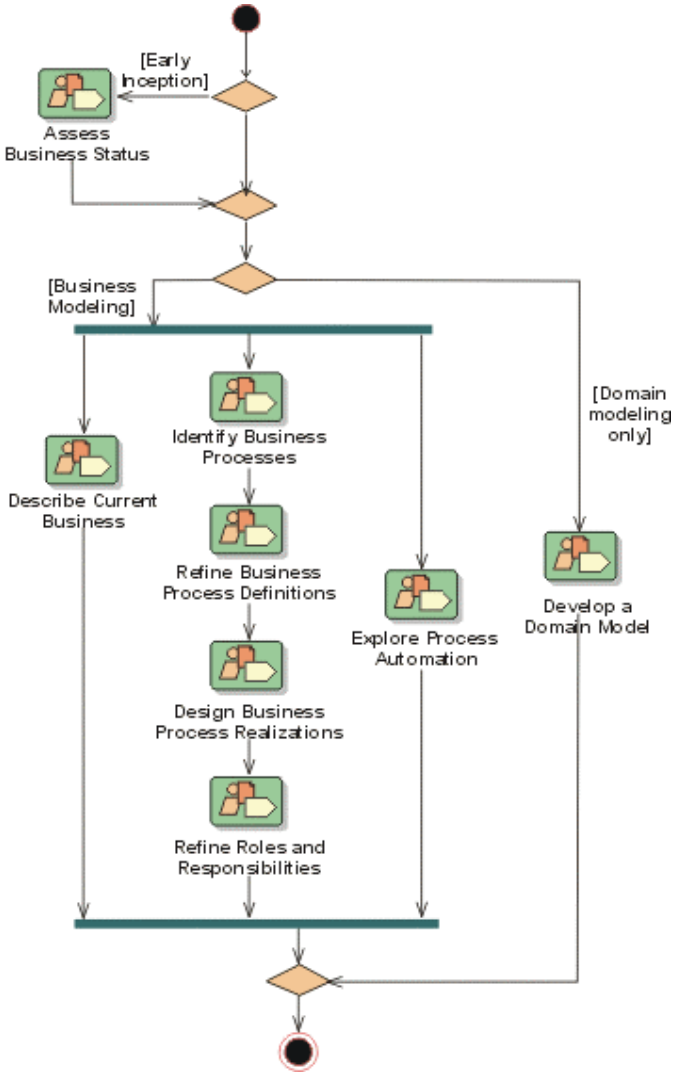
Business Modeling Workflow
Syftet med arbetsflödet är

- att förstå strukturen och dynamiken hos organisationen i vilken systemet ska driftsättas (målorganisationen)
- att förstå nuvarande problem i målorganisationen och identifiera förbättringsmöjligheter
- att säkerställa att kunder, slutanvändare och utvecklare har en gemensam förståelse för målorganisationen
- att härleda de systemkrav som behövs för att stödja målorganisationen (Kruchten, 2002, s 139)

Det finns två huvudroller vid verksamhetsmodelleringen: verksamhetsanalytiker (*Business-Process Analyst*) och verksamhetsutvecklare (*Business Designer*). Den första styr och samordnar modelleringen av användningsfallen för verksamheten och den andra beskriver användningsfallen. Dessutom finns det en verksamhetsgranskare (*Business Model Reviewer*) som granskar resulterande artefakter. Nyckelartefakterna i arbetsflödet är *Business Vision* som definierar målen för arbetet, *Business Use-Case Model* som är en modell över verksamhetens funktioner, samt *Business Object Model* som beskriver hur verksamhetens användningsfall realiseras.

Omfattningen av arbetet kan variera beroende på sammanhanget. Exempel på scenarier vid verksamhetsmodellering är

- att skapa en organisationskarta för att få en förståelse för kraven på den applikation som ska byggas,
- skapa en allmängiltig verksamhetsmodell som kan användas av flera organisationer samt
- att bygga stödjande informationssystem för organisationer som vill starta en verksamhetslinje. (Kruchten, 2002, s 143-144).



Figur 3.5.1.2 *Business Modeling Workflow*

Arbetsflödesdetalj	**Aktivitet**
Assess Business Status	Capture a Common Vocabulary Maintain Business Rules Assess Target Organization Set and Adjust Goals
Describe Current Business	Assess Target Organization Find Business Actors and Use Cases Set and Adjust Goals Find Business Workers and Entities
Identify Business Processes	Maintain Business Rules Set and Adjust Goals Define Business Architecture Capture a Common Business Vocabulary Find Business Actors and Use Cases
Refine Business Process Definitions	Structure the Business Use-Case Model Detail Business Use Case Review the Business Use-Case Model

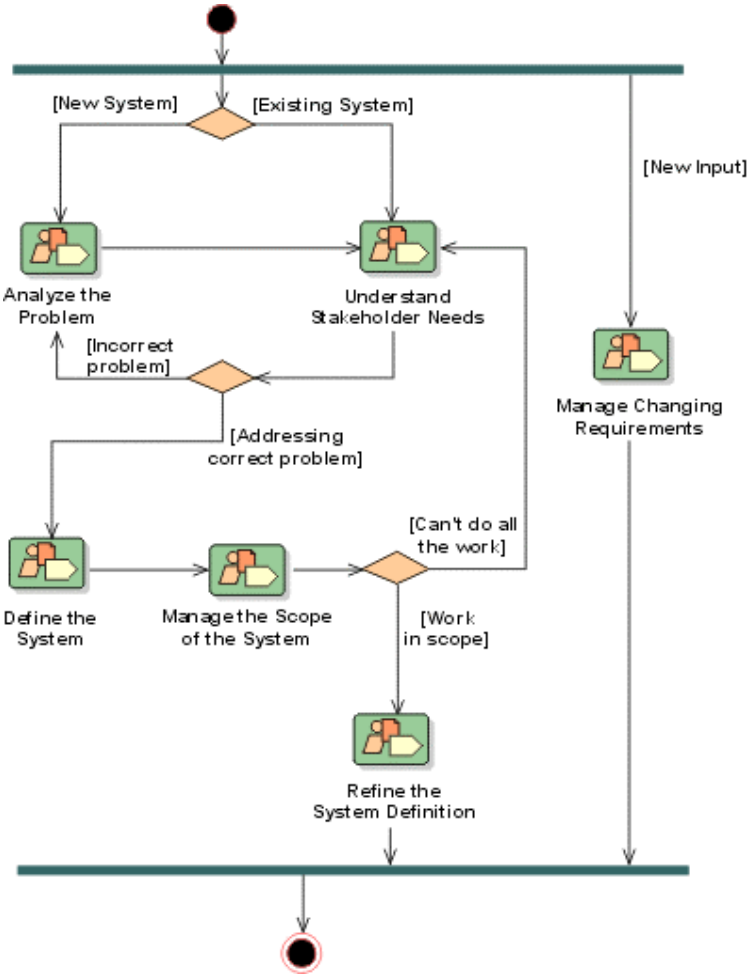
Design Business Process Realizations	Find Business Workers and Entities Capture Common Business Vocabulary Maintain Business Rules Define Business Architecture
Refine Roles and Responsibilities	Detail a Business Worker Detail Business Entity Review Business Object Model
Explore Process Automation	Set and Adjust Goals Define Automation Requirements
Develop a Domain Model	Maintain Business Rules Capture a Common Business Vocabulary Find Business Workers and Entities Detail a Business Entity Review the Business Object Model

Tabell 3.5.1.2 *Aktiviteterna inom respektive arbetsflödesdetalj i Business Modeling Workflow*

Requirements Workflow
Målet för arbetsflödet är att

- fastställa och upprätthålla en överenskommelse med kunder och andra intressenter om vad systemet ska göra – och varför!
- ge systemutvecklare en förståelse för systemkraven
- definiera systemets gränser
- ge grunden för planeringen av det tekniska innehållet i iterationerna
- ge grunden för att beräkna kostnader och tid för att utveckla systemet
- definiera ett användargränssnitt till systemet som fokuserar på användarnas behov och mål (Kruchten, 2002, s 157)

Kraven samlas in via *Stakeholder Requests* och utgör en ”önskelista” från kunden. De viktigaste högnivåkraven samlas sedan i ett Vision-dokument som ska skrivas utifrån kundens perspektiv. Vision-dokumentet är viktigt, eftersom det är basen till kontraktet med kunden. Alla krav som kunden kan se (features) ska räknas upp. I *Software Requirements Specification* ingår användningsfallsmodellen, som ibland är kontraktet med kunden, och *Supplementary Specification* för att täcka in både funktionella och ickefunktionella krav. Det mesta arbetet utförs av en systemanalytiker (*System Analyst*), men användningsfallsspecificerare (*Use-Case Specifier)* finns med för att detaljera delar av systemets funktionalitet. En särskild användargränssnittsdesigner (*User-Interface Designer*) finns också, liksom en kravgranskare (*Requirements Reviewer*), som egentligen är vem som helst som är med och verifierar att utvecklingsgruppen har tolkat kraven rätt.



Figur 3.5.1.3 *Requirements Workflow*

Arbetsflödesdetalj	**Aktivitet**
Analyze the Problem	Capture a Common Vocabulary Develop Requirements Management Plan Find Actors and Use Cases Develop Vision
Understand Stakeholder Needs	Capture a Common Vocabulary Develop Vision Manage Dependencies Elicit Stakeholder Requests Find Actors and Use Cases
Define the System	Develop Vision Manage Dependencies Capture Common Vocabulary Find Actors and Use Cases
Manage the Scope of the System	Prioritize Use Cases Develop Vision Manage Dependencies
Refine the System Definition	Detail a Use Case Detail the Software Requirements Model the User-Interface

	Prototype the User-Interface
Manage Changing Requirements	Structure the Use-Case Model
	Manage Dependencies
	Review Requirements

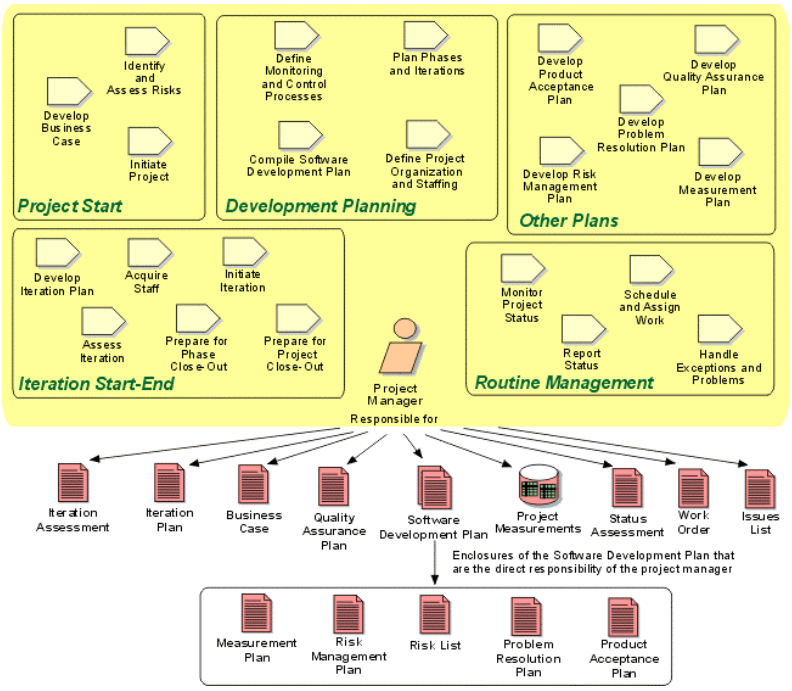
Tabell 3.5.1.3 *Aktiviteterna inom respektive arbetsflödesdetalj i Project Management Workflow*

3.5.2 Roller

En *roll* beskriver ansvaret hos en person eller en grupp personer och definierar *vem* som ska utföra arbetet. Den kan innehas av olika projektmedlemmar, som i sin tur kan ha flera olika roller i samma projekt. Exempel på roller är systemanalytiker, designer och projektledare.

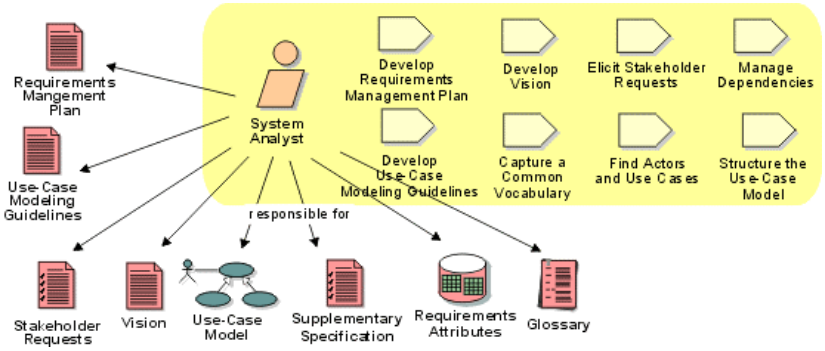
De roller som kommer att beröras i examensarbetet beskrivs här kortfattat:

Project Manager: I RUP är det projektledaren (*Project Manager*) som styr hela utvecklingsprojektet. Han eller hon allokerar resurser, prioriterar arbetet och koordinerar kontakterna med kunder och användare. Projektledaren ska också skapa rutiner för att projektets dokument håller hög kvalitet. I figur 3.5.7 finns en översikt över de aktiviteter och de dokument som projektledaren har ansvar för. Det är inte mindre än 14 dokument som ska skapas varav fem ingår i *Software Development Plan*. Allt som ska planeras dokumenteras.



Figur 3.5.2.1 *Rollen Project Manager.*

System Analyst: Kravinsamling och modellering av användningsfall är systemanalytikerns ansvar. Det är också systemanalytikern som bestämmer vilka aktörer och användningsfall som får finnas och därmed också systemets omfattning. De dokument som systemanalytikern är ansvarig för är främst *Vision* och de dokument som hanterar användningsfall och andra krav.

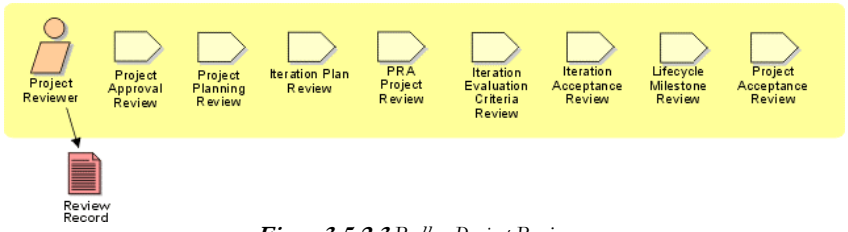


Figur 3.5.2.2 *Rollen System Analyst..*

Vad systemet ska åstadkomma kommer helt från intressentönskemålen (*Stakeholder Request*). Därefter definieras aktörer och användningsfall.

Stakeholder: en intressent är den som har ett materiellt intresse av hur produkten blir. Olika intressenter är kunder och användare eller deras representanter; de som har ett ekonomiskt intresse av hur marknaden för produkten blir som investerare och aktieägare; representanter från utvecklingssidan som testare, designers, dokumentationsskribenter osv. Intressenter har inget eget ansvar för några dokument, utan finns med som stödjande roller vid olika aktiviteter som vid kravinsamling och avgränsning av systemet. Typiskt i RUP är att kund *och* användare *och* en allmän intressent är explicit angivna som stödjande roller om intressenter ska vara med.

Project Reviewer: Projektgranskaren har stort inflytande över projektets framåtskridande. Vid åtta punkter i projektet granskas olika artefakter och utvärderingar av projektet. Vid dessa granskningstillfällen kan projektet läggas ner om projektet inte är tillräckligt välplanerat eller om framåtskridandet är för dåligt.



Figur 3.5.2.3 *Rollen Project Reviewer.*

Software Architect: Programvaruarkitekten är bara delvis inblandad i arbetsflödet för *Requirements* och inte alls i *Project Management* eller *Business Modeling*. I *Requirements* är det programvaruarkiteten som prioriterar vilka användningsfall som ska implementeras i en viss iteration. Då väljs i första hand användningsfall med central funktionalitet eller användningsfall som täcker en stor del av den nödvändigaste arkitekturen.

PRA (Project Review Authority): PRA är egentligen ingen egen roll utan mer en organisatorisk instans som projektledaren rapporterar till löpande under projektets gång (PRA Project Review). PRA är ansvarigt för policies, standarder och praxis för organisationen och ser till att kontraktet följs, men också att kunden sköter sina åtaganden gentemot den utvecklande organisationen. Rational rekommenderar att en enskild person utses till PRA, och som kan begära in stöd av andra för beslutsfattandet.

3.5.3 Aktiviteter

Aktiviteter definierar *hur* en uppgift ska utföras och resulterar i att en eller flera artefakter skapas eller uppdateras. Varje aktivitet tilldelas och utförs oftast av en enda roll, men kan få stöd från andra roller.

3.5.4 Artefakter

Artefakter redogör *vad* som ska utföras och är det som används och produceras i aktiviteter. Det är alltså det konkreta resultatet av projektet och är olika typer av material som t ex en modell eller modellelement, dokument, källkod och exekverbar kod.

Kapitel 4

Metod

Examensarbetet bestod av två huvudsakliga delar: 1. analys av RUP och 2. utveckling av förslag till MRUP (*Market driven Rational Unified Process*). Metodiken som användes hör hemma bland ingenjörsmetoderna (*Engineering method),* enligt en klassificering som utvecklades vid en workshop 1989 (Adrion, 1993). Ingenjörsmetoder innebär att existerande lösningar först studeras och att ändringar sedan föreslås som analyseras. Detta förfarande upprepas tills lösningen har uppnått en tillräckligt bra nivå. Examensarbetets analys motsvarar studiet av de existerande lösningarna. De specifika metoderna som användes redovisas separat under 4.1 och 4.2.

Endast de inledande delarna av ett projekt som berörs av kravhanteringen analyserades, d v s *Project Management Workflow*, *Business Modeling Workflow* och *Requirements Workflow*. Resterande arbetsflöden berör arbetssättet vid själva utvecklingen, vilket troligen inte påverkas nämnvärt, mer än att dokument som används kommer att ha något förändrat innehåll.

4.1 Analys

Syftet med analysen var att fastställa hur de existerande lösningarna på kravhanteringen såg ut. Specifikt handlade det om att finna kontraktsdrivna element som kunde lägga hinder i vägen mot eventuella försök att använda dagens RUP på ett marknadsdrivet sätt. Analysen genomfördes i två steg: 1. ett datainsamlingssteg och 2. ett fördjupande kvalitativt analyssteg.

Metoden som användes för datainsamling var en form av innehållsanalys (c*ontent analysis).* Innehållsanalys är analys av befintlig dokumentation där t ex förekomsten av vissa ord kan registreras (Robson, 2002). Fördelarna med innehållsanalys är att dokumenten inte påverkas. Det är alltså möjligt att göra om datainsamlingen med samma resultat. Nackdelarna är att dokumenten inte ger hela sanningen och att de skrivits för andra syften än forskningen.

RUP innehåller beskrivande text som var lämplig för innehållsanalys av vad som antagits vid skapandet av processen. Med tabell 3.3.1 som utgångspunkt analyserades textmassan i *Project Management Workflow, Requirements Workflow* och *Business Modeling Workflow*. Allt som explicit gick att härröra till egenskaper som hör till kontraktsdriven utveckling noterades. Materialet gicks igenom tre gånger för att inget skulle missas.

Enligt Berelson (1952) är en innehållsanalys inte bättre än dess kategorier. Kategorierna som användes för examensarbetet var begreppen i tabell 3.3.1. Några begrepp uteslöts av olika skäl:

1. *Fysiskt avstånd till användaren*, eftersom användaren behandlades under begreppet användare.
2. *Utvecklarnas anknytning till programvaran*, eftersom det inte gick att utläsa ur materialet.
3. *Kravhanteringsstandarder*, eftersom RUP i sig standardiserar detta arbete.

4. *Iterativa tekniker*, eftersom RUP i sig förespråkar ett iterativt arbetssätt.
5. *Domänexperter tillgängliga bland utvecklarna*, eftersom det inte gick att utläsa ur materialet.

Begreppen registrerades efter förekomst i en viss aktivitet eller artefakt. Förekomsten av ett visst begrepp noterades i en matris, som representerar en kvantifiering av innehållsanalysen. Där framgår den relativa utbredningen av de olika begreppen och redovisas i kapitel 5.1. Hela citat eller brottsstycken av texten sparades separat för senare kvalitativ analys, som redovisas i kapitel 5.2.

Här följer några exempel på hur begreppen tolkades:

> "The customer representative agrees that the project has met expectations for the phase".

Meningen hittades i aktiviteten *Lifecycle Milestone Review*. Kunden är uttalat kontraktsinriktad och aktiviteten markeras som kontraktsinriktad i egenskapen kund. Framgångsmåttet är att kunden är nöjd vilket gör att också denna egenskap blir kontraktsinriktad.

> "The contract may also allow the customer to request audits."

Meningen hittades i aktiviteten *Develop Quality Assurance Plan* och gjorde att både kund och kontrakt noterades som kontraktsinriktade.

Alla aktiviteter och artefakter, inklusive riktlinjer, kontrollpunkter och begrepp behandlades på detta sätt. För att en aktivitet eller artefakt skulle tolkas som kontraktsdriven krävdes att den specifika egenskapen var direkt uttalad och inte gick att välja bort, t ex om det gavs ett marknadsinriktat alternativt handlingssätt. På så sätt kunde risken för subjektiva tolkningar minskas.

Motsvarande genomgång gjordes för marknadsdrivna egenskaper, men det krävdes enbart att det var *möjligt* att använda artefakten eller aktiviteten marknadsdrivet med avseende på en viss egenskap. Ett exempel på en sådan konstruktion är:

> "If the purpose of your project is to build a generic, customizable application (such as a commercial accounting application)…"

Meningen hittades i aktiviteten *Set and Adjust Goals*. Eftersom systemmodellering och kund kunde tolkas marknadsdrivet noterades dessa egenskaper som marknadsdrivna.

Gråzoner där aktiviteterna eller artefakterna varken kunde tolkas som kontraktsdrivna eller marknadsdrivna har inte markerats i matriserna.

I det andra fördjupande kvalitativa analyssteget diskuteras hur kontraktsinriktade begreppen påverkar möjligheterna att använda RUP marknadsdrivet. Där tas frågor som inte täcks in av analysverktyget upp, som timing av olika aktiviteter och frågor som blir för stora för analysverktyget. Det gäller t ex validering och specifika kravhanteringsaktiviteter. Aktivitetsstöd som borde finnas vid marknadsinriktad utveckling behandlas där det är relevant. För att inte riskera upprepningar har vissa kategorier slagits samman.

4.2 Utveckling av förslag till MRUP

Vid utvecklingen av ett förslag till MRUP (*Market driven Rational Unified Process)* var det viktigt att överbrygga avståndet till möjliga användare för de nya arbetssätten. Utveckling av förslaget varvades med utvärderingar via fokusgruppmöten för att fånga in synpunkter som bidrog till vidareutveckling av förslaget. Där kontrollerades också att arbetssätten blev tydligt beskrivna och relevanta. Den här typen av utvärderingar kallas *formativa utvärderingar*, eftersom utvärderingen i sig bidrar till utvecklingen av resultatet (Robson, 2002, s 206; Trochim, 2003).

Arbetet med nya arbetsflöden bedrevs på bred front. Med ett öppet sinne till vad som kunde behöva integreras i en ny modell lades fokus på bredd, snarare än djup, för att inte missa några viktiga aktiviteter. Att i detalj utarbeta aktiviteterna med beskrivningar fick anses ligga utanför detta examensarbetes ramar, särskilt eftersom de delar som utförs av marknadspersonal borde genomarbetas av marknadsekonomer. Modellen bör också genomgå testning i verkliga projekt i större skala. Det var inte okomplicerat att fastställa *vilka* större aktiviteter som skulle genomföras och inom kravhanteringsområdet finns det tyvärr inga givna regler om *hur* de föreslagna aktiviteterna sedan bör utföras. Den bakgrund som presenterats i teorikapitlet får fungera som bakgrund till hur det mer detaljerade arbetet kan se ut.

Utgångspunkten till förslagen till alternativa arbetsflöden var vad som innehållsmässigt talade emot marknadsdriven utveckling i RUPs befintliga skick. Från den industrinära forskning på problemområden för marknadsdriven utveckling som bedrivs hämtades relevanta kompletterande aktiviteter. Dessutom användes de aktiviteter som är centrala i produktutvecklingsteorier för att göra ett förslag till en mer marknadsorienterad RUP, MRUP.

Programvaruindustrin arbetar ingalunda på ett homogent sätt. Det gjorde att förslaget utformades på ett allmänt sätt utan detaljer som låser fast företaget vid ett visst handlingssätt. Däremot ges vissa rekommendationer till hur aktiviteterna ska genomföras.

En första modell utvecklades först med ingående roller, arbetsflöden, aktiviteter och exempel på centrala dokument som antogs behövas. Denna modell genomgick sedan ytterligare fem iterationer med utvärderingar och vidareutveckling från industri- och universitetsrepresentanter däremellan. På grund av tidsramarna utarbetades modellen på ett ganska övergripande plan, eftersom detaljnivån skulle krävt noggrannare tester och vidareutveckling i verkliga projekt.

Deltagare till fokusgruppmötena valdes efter sin erfarenhet om kravhanteringsfrågor och marknadsdriven programvaruutveckling. Några av dem hade också erfarenhet av att utforma utvecklingsprocesser. Det fanns också en spridning i typ av arbetsplatser de hade erfarenheter från: små-mellanstora företag, stora företag samt universitetet. Företagsrepresentanterna varierade i hur nära produktledarrollen respektive projektledarrollen de arbetade. Deltagarna tilldelades material i förväg, samt försågs med ett antal nyckelproblem som varje iteration fokuserade på.

Iteration	Deltagare	Erfarenhetsområden representerade	Nyckelproblem
1	En konsult (stora och mellanstora företag) En doktorand i kravhantering	Produkt/projektledning Marknadsdriven kravhantering	Omfattningen av *Project Management Workflow* Godkännande av planering Maturity Reviews (godkännande inför lansering) Beslutspunkter Kontroll över *Vision* Nedläggning av utvecklingsprojekt
2	Två industrirepresentanter (mellanstora företag)	Produkt/projektledning Processutveckling Kravhantering	Omfattningen av *Project Management Workflow* Beslutspunkter Nödvändiga granskningar Marknadsavdelningens roll COTS-beslut[24] Utformningen av *Vision*
3	En senior forskare inom kravhantering	Kravhantering Processutveckling	Relationer mellan beroendeanalys – kostnadsanalys – påverkansanalys Beslutspunkter Gränssnitt strategiskt arbete och utvecklingsarbete Utformningen av *Vision*
4	Två konsulter (mellanstora och stora företag)	Kravhantering Processanpassningar RUP	Projektintegration Maturity Reviews Gränssnitt mot projektarbete Beslutspunkter Nödvändiga dokument i *Project Management Workflow*
5	Två industrirepresentanter (stora företag)	Kravhantering Produktledning	Projektintegration Maturity Reviews Gränssnitt mot projektarbete Storskaliga projekt

Tabell 4.2.1 *Översikt över utvärderingarna av förslagets iterationer.*

Förslaget med de modifierade arbetsflödena och ingående arbetsflödesdetaljer, aktiviteter och roller framställdes som bildmaterial. Detta gjordes med hjälp av Rationals Workbench. Bildmaterialet kompletterades med kortfattade beskrivningar i punktform över vad varje aktivitet bör inbegripa, samt motiveringar till aktiviteterna. Därutöver omformades två centrala malldokument, *Vision* och *Business Case* för att stödja de modifierade arbetsflödena.

[24] COTS betyder Commersial Off The Shelf och används om kommersiella programvarukomponenter som integreras med den egna produkten. Anses vara ett sätt att effektivisera utvecklingen.

4.3 Validitet

Validitet brukar ofta missuppfattas och förknippas med olika beståndsdelar inom forskningen, som t ex mätvärden, stickprov eller design. Man kan exempelvis prata om giltiga mätvärden. Men mätvärden i sig har ingen validitet – det är påståenden och slutsatser som har det (Trochim, 2002). Det finns fyra olika nivåer av validitet som kan diskuteras med hjälp av vars en fråga:

- Är slutsatserna från analysen giltiga (*conclusion validity*)?
  Analysen av RUP genomfördes i två steg, varav det första genomfördes tre gånger för att säkerställa att inget har missats (se kap 4.1). I det andra steget gjordes en djupare analys för att få med kvaliteter som inte kunde täckas in av det första steget. Analysen var nödvändig för att ha en bra utgångspunkt vid utvecklingen av förslaget till MRUP. Hot mot denna validitetsnivå kan vara analysverktyget (tabell 3.3.1), som kan anses vara otillräckligt och ej uttömmande. Men tabellen användes ändå för att ha något att utgå ifrån.

- Är resultatet giltigt (*internal validity*)?
  Man kan alltid ifrågasätta om ett förslag som två studenter, utan erfarenhet från promgramvaruutvecklingsprojekt, har utvecklat är giltigt eller inte. För att få ett giltigt resultat och för att modellen ska bli accepterad av industrin har erfarenheter från både universitetsvärlden och industrin lånats och använts i de formativa utvärderingarna (se kap 4.2).

- Har vi löst rätt problem (*construct validity*)?
  Eftersom modellen har utvecklats med hjälp av formativa utvärderingar skulle man kunna säga att man har löst rätt problem för de som deltog i utvärderingarna. Men frågan är om alla företag har samma problem och om modellen är fullständig? Detta kan bara besvaras genom att testa modellen i verkliga projekt.

- Är resultatet generellt applicerbart (*external validity*)?
  För att få en generell lösning på den nya modellen genomfördes utvärderingarna med hjälp av folk från olika stora företag med varierande bakgrund (se kap 4.2). Hot mot denna validitetsnivå kan vara att deltagarna till fokusgruppmötena endast kunde ge input på problem som de själva stött på och förklara hur de hade löst dem. Det bästa sättet att komma ifrån detta är att testa modellen i verkliga projekt på olika företag.

Kapitel 5

Analysresultat

Resultatet presenteras i olika steg för att belysa olika problemaspekter hos RUP. Först redovisas den mer övergripande kvantifieringen av analysen som visar vilka delar av RUP som har mer eller mindre kontraktsdrivna drag och där eventuella inkonsekvenser träder fram. En kvalitativ genomgång av kontraktsdrivna karaktäristika görs sedan, där tonvikten ligger på hur utbredda de är i centrala dokument och hur de försvårar marknadsdriven utveckling. Karaktärsdrag som är särskilt inriktade mot kontraktsdriven utveckling ringas in och diskuteras i relation till en marknadsdriven motsvarighet. Problem som har med timing och rollfördelning att göra tas upp där det är relevant.

5.1 Kvantifiering av innehållsanalysen per arbetsflöde

Kvantifieringen av analysen presenteras i tabellform. Alla kontraktsdrivna egenskaper har markerats med 'o' och alla marknadsdrivna med 'x'. Arbetsflödena *Project Management Workflow*, *Business Modeling Workflow* och *Requirements Workflow* redovisas separat. Med hjälp av tabellerna kan man få en överblick över de aktiviteter och artefakter som använder sig av kontraktsbaserade och marknadsdrivna begrepp, och även se skillnader mellan arbetsflödena. Det bör poängteras att tabellerna inte är kompletta eftersom endast det som är explicit uttryckt är representerat. Syftet med tabellerna är inte att visa alla detaljer, utan bara var problemen kan finnas. Tabellerna är användbara som stöd för läsaren vid den kvalitativa analysen.

5.1.1 Project Management Workflow

Project Management Workflow	Primärt mål	Framgångsmått	Livscykel	Kravens ursprung	Kravspecifikation	Användare	Kund	Huvudintressent	Speciella kravhanteringsaktiviteter	Validering	Systemmodellering	Konkurrenter	Val av innehåll, leverantid och prissättning
Aktiviteter													
Project Approval Review		x											
Identify and Assess Risks													
Develop Business Case	x	x	x				x	x				x	
Initiate Project		o	o				o						
Develop Measurement Plan	x												
Develop Risk Management Plan													
Develop Product Acceptance Plan		o	o				o						
Develop Problem Resolution Plan													
Develop Quality Assurance Plan							o						
Define Project Organization and Staffing													
Define Monitoring and Control Processes													
Plan Phases and Iterations		o	o				o						
Compile Software Development Plan													
Project Planning Review													
Iteration Plan Review							o						
Develop Iteration Plan													
Iteration Evaluation Criteria Review		o					o						
Iteration Acceptance Review		o					o						
Acquire Staff													
Initiate Iteration													
Assess Iteration												x	x
Project Acceptance Review	o	o					o						
Prepare for Project Close-Out		o	o				o						
Lifecycle Milestone Review		o					o						
Prepare for Phase Close-Out		o					o						
PRA Project Review													
Monitor Project Status													
Report Status													
Schedule and Assign Work													
Handle Exceptions and Problems							o						
Artekfater													
Software Development Plan							o						
Business Case		x											
Iteration Plan													
Iteration Assessment													
Status Assessment													
Problem Resolution Plan													
Risk Management Plan													
Risk List	x											x	
Work Order													
Product Acceptance Plan		o					o						
Measurement Plan							o						
Quality Assurance Plan	o	o					o						
Issues List													
Project Measurements													
Review Record							o						

Tabell 5.1.1.1 *Kvantifiering av innehållsanalysen av Project Management Workflow.*
o = strikt kontraktsdriven egenskap, x = marknadsdriven egenskap möjlig

Mycket av det som utförs och skrivs i *Project Management Workflow* har typiska kontraktsbaserade karaktäristika och inga antydningar till marknadsdriven utveckling. De marknadsinriktade egenskaperna är begränsade till vissa specifika aktiviteter och artefakter. Att det finns en specifik kund är centralt redan vid uppstarten av projektet i aktiviteten *Initiate Project* och många av aktiviteterna är till för att få godkännande och se till att kontraktet följs. Alla granskningsaktiviteter, *PRA Project Review* undantaget, refererar till en specifik kund, liksom andra aktiviteter som har med acceptans och evaluering att göra[25]. Likaså är planering och uppföljning av faser och iterationer kopplade till en beställare[26], och aktiviteter och artefakter som är till för att skapa allmän insyn, som *Measurement Plan* och *Quality Assurance Plan*. Signifikant är också att referenser till en kund går hand i hand med referenser till ett kontrakt.

Många av aktiviteterna och artefakterna är allmänt hållna och innehåller varken marknads- eller kontraktsinriktade begrepp. Det gäller alla dokument som inte fått några markeringar alls. En anledning kan vara att just de aktiviteterna och dokumenten är oberoende av hur det ekonomiska intresset ser ut, men troligare är att RUP helt enkelt rensats i dessa delar, och att den utvecklande organisationen ska välja lämpliga aktiviteter och artefakter själv.

Endast ett fåtal aktiviteter och artefakter i *Project Management Workflow* har direkta hänvisningar till marknadsdriven utveckling, och särskilt utmärker sig aktiviteten *Develop Business Case* med resulterande artefakt *Business Case*. Från ett mycket sparsamt bruk av marknadstermer, är denna aktivitet närmast ensam om att tydligt markera möjligheten till marknadsmässig utveckling. Därutöver rekommenderar endast aktiviteten *Assess Iteration* omvärldsanalys och hänsyn till konkurrenters agerande på marknaden och till artefakten *Risk List* rekommenderas att affärsrisker inkluderas.

I det här arbetsflödet är det alltså särskilt aktiviteterna som är kopplade till kontraktsdriven utveckling. Även om de kontraktsbaserade aktiviteterna tas bort kommer stöd för marknadsdrivna aktiviteter som t ex releaseplanering att saknas. I den kvalitativa analysen undersöks hur den stora mängden av dokument och deras relevans är drabbad av kontrakts- och kundinriktningen.

[25] *Develop Product Acceptance Plan* med resulterande artefakt *Product Acceptance Plan; Iteration Evaluation Criteria Review*, *Iteration Acceptance Review*.

[26] *Plan Phases and Iterations*, *Project Acceptance Review*, *Prepare for Project Close-Out*, *Lifecycle Milestone Review*, *Prepare for Phase Close-Out*.

5.1.2 Business Modeling Workflow

Business Modeling Workflow	Primärt mål	Framgångsmått	Livscykel	Kravens ursprung	Kravspecification	Användare	Kund	Huvudintressent	Speciella kravhanteringsaktiviteter	Validering	Systemmodellering	Konkurrenter	Val av innehåll, leveranstid och prissättning
Aktiviteter													
Capture a Common Business Vocabulary						o	o						
Maintain Business Rules						o	o						
Assess Target Organization						o	o				o		
Set and Adjust Goals						o	ox				ox		
Find Business Actors and Use Cases						o	o						
Find Business Workers and Entities						o	o						
Define Business Architecture						o	o						
Structure the Business Use-Case Model						o	o						
Review the Business Use-Case Model						o	o						
Detail a Business Use-Case						o	o						
Detail a Business Worker						o	o						
Detail a Business Entity						o	o						
Review the Business Object Model						o	o						
Define Automation Requirements						o	o				o		
Artefakter													
Business Glossary													
Business Rules													
Business Use-Case Model							o				o		
Business Object Model													
Target Organization Assessment							o				o		
Business Vision							o				o		
Business Architecture Document											o		
Supplementary Business Specification													
Business Use Case							o				o		
Organization Unit							o				o		
Business Actor							o				o		
Business Entity							o				o		
Business Use-Case Realization							o				o		
Business Worker							o				o		

Tabell 5.1.2.1 *Kvantifiering av innehållsanalysen av Business Modeling Workflow.*
o = strikt kontraktsdriven egenskap, x = marknadsdriven egenskap möjlig

Arbetsflödet *Business Modeling Workflow* är konsekvent inriktat på en specifik kund. Det är inte så konstigt, eftersom det uttalat är en specifik kundorganisation som modelleras i RUP. För att kunna göra en bra modellering av organisationen måste även användarna vara identifierbara och involverade när aktiviteterna utförs, men syns inte i de resulterande artefakterna. Modelleringen av kundorganisationens system beskrivs mer i detalj vid utvecklingen av artefakterna, vilket tydligt syns i tabellen.

Syftet med *Business Modeling Workflow* är kontraktsdrivet, men arbetsflödet går relativt enkelt att anpassa och skulle även kunna användas i marknadsdriven utveckling. Det är också anledningen till att *Set and Adjust Goals* har markerats som delvis marknadsdriven. Arbetsflödet kan nämligen användas till att bygga en allmängiltig anpassningsbar applikation (t ex ett affärssystem). En verklig eller idealiserad kundorganisation skulle kunna utnyttjas för att skapa en generell modell som skulle kunna appliceras på andra organisationer. Då modifierar man helt enkelt *Assess Target Organization* för att skaffa sig en bild över hur branschens arbetssätt i stort ser ut. En skillnad blir då att de tekniska kraven

inte blir lika specifika, utan en viss flexibilitet får byggas in. Eftersom arbetsflödet troligen är så pass enkelt att justera kommer det inte att ingå vid utvecklingen av en modifiering för marknadsdriven utveckling.

5.1.3 Requirements Workflow

Requirements Workflow	Primärt mål	Framgångsmått	Livscykel	Kravens ursprung	Kravspecification	Användare	Kund	Huvudintressent	Speciella kravhanteringsaktiviteter	Validering	Systemmodellering	Konkurrenter	Val av innehåll, leverans och prissättning
Aktiviteter													
Capture a Common Vocabulary											o		
Develop Requirements Management Plan													
Find Actors and Use Cases											o		
Develop Vision													x
Elicit Stakeholder Requests			x	o			x						
Manage Dependencies													
Prioritize Use Cases													
Detail a Use Case						o	o						
Detail the Software Requirements													
Structure the Use-Case Model						o	o						
Review Requirements				o			o			o			
Model the User-Interface													
Prototype the User-Interface													
Artefakter													
Requirements Management Plan	x	x	x				x						
Stakeholder Requests			o				x		ox		o		
Vision			x		o		x	x	x				x
Glossary													
Use-Case Model					o	o	o	o			o		o
Supplementary Specifications							o						
Requirements Attributes			x										
Use Case					o	o	o	o					o
Software Requirements Specification	o			o	o	o	o				o		
Use-Case Package													
User-Interface Prototype													
Use-Case Storyboard													

Tabell 5.1.3.1 *Kvantifiering av innehållsanalysen av Requirements Workflow.*
o = strikt kontraktsdriven egenskap, x = marknadsdriven egenskap möjlig

Tabell 5.1.3.1 visar alla aktiviteter och artefakter som finns i *Requirements Workflow*. Det första intrycket är att begreppsbilden är ganska splittrad. Det är dock vanligast med kontraktsinriktade begrepp och då är det främst artefakterna som har begreppsmässiga kopplingar till antingen marknads- eller kontraktsbegrepp. Ett fåtal artefakter och aktiviteter använder sig av marknadstermer, men det kan samtidigt finnas hänvisningar till kontraktsdrivna egenskaper.

Några av aktiviteterna som hanterar utvecklingen av användningsfall[27] utgår ifrån att kunden eller användarna är identifierbara och involverade, vilket är svårare i marknadsdrivna projekt. Användarna är följaktligen också engagerade i artefakterna som har med användningsfall att göra. Dessutom fungerar artefakten *Vision* som kontraktsbas

[27] *Detail a Use Case*, *Structure the Use-Case Model* och *Review Requirements*.

och kompletteras med de mer tekniskt detaljerade artefakterna *Use Case, Use-Case Model* och *Software Requirements Specification*. Dessa är inte heller relevanta när produkten utvecklas för en marknad, även om artefakterna i sig kan fylla andra syften. Av tabell 5.1.3.1 framgår också att utgångspunkten för systemmodelleringen är de befintliga strukturerna hos kunden, till exempel i artefakten *Use-Case Model* och i de initiala kraven i form av artefakten *Stakeholder Requests*.

Det märks tydligt att det har gjorts en ansats för att göra RUP mer lämplig för marknadsdriven utveckling. Flera aktiviteter och artefakter har gjort sig förtjänta av en del kryss medan andra varken har kontraktsbaserade eller marknadsdrivna drag, vilket egentligen inte är så konstigt. Aktiviteter som *Manage Dependencies* och *Prioritize Use Cases* ska utföras oberoende om produkten utvecklas för en specifik kund eller för en marknad.

Formuleringar som inte utesluter marknadsdriven utveckling har framförallt lagts till i artefakterna *Requirements Management Plan, Stakeholder Request* och *Vision*. Samtidigt finns det inte lika många referenser till ett kontraktsdrivet arbetssätt. Framförallt utesluter artefakterna inte möjligheten till en allmän kund på marknaden och mer marknadsmässiga tänkande i form av prissättning, licensieringsfrågor och försäljning kan finnas med i det centrala dokumentet *Vision*. Malldokumentet för *Vision* lider av missuppfattningar om termer som t ex positionering. I dokumentet fungerar positionering som rubrik för information om vilket problem som produkten ska lösa, vilken affärsmöjlighet som öppnar sig och om vad företaget vill åstadkomma med sin produkt på marknaden. Å andra sidan handlar kapitlet om marknadsdemografi bland annat om sådant som har med positionering att göra.

De uppenbart marknadsinriktade delarna ihop med de mer allmänna skulle lite välvilligt kunna tolkas som att arbetsflödet *Requirements* går att använda till ett marknadsdrivet arbetssätt, om det inte saknats visst aktivitetsstöd för t ex releaseplanering. Den kvalitativa analysen kommer att visa att kopplingen mellan arbetsflödets dokument och kontraktsdriven utveckling faktiskt inte är fullt så hård som tabellen antyder. Dokumenten har kvar sin relevans även utan ett kontrakt.

5.2 Kvalitativ analys per problemområde

5.2.1 Huvudintressent

En intressent (*stakeholder*) definieras i RUP som ”an individual who is materially affected by the outcome of the system” (ur RUPs *glossary*). I den faktiska användningen av termen i artefakter och aktiviteter är betydelsen dock inte lika vid. I introduktionen till artefakten *Stakeholder Request* är stakeholder “a customer, end user, marketing person, and so on”, d v s fokus ligger på kund och slutanvändare. Utvecklarna är inte explicit nämnda och avses troligen inte.

I *Stakeholder Request* talar man om intressenter och användare (*end user*). Informationen som fås från intressenten rör vilka problem med existerande lösningar som han eller hon anser finnas och hur det löses idag. Intressenten ska också ge respons på problem som systemanalytikern tycker borde röra honom/henne. Det medför att systemanalytikern inte anses vara en intressent. Det framgår också av dokumentmallen för *Vision* att intressenten har behov (*needs*) och att Visiondokumentet ska fokusera på funktionalitet som behövs av intressenten och användarna. Eftersom användaren inte räknas som intressent blir slutsatsen att huvudintressenten faktiskt är kunden själv.

I RUP talar man aldrig om huvudintressent (*main stakeholder*). Antingen finns det en uttalad kund eller så finns det utrymme för att utveckla för en öppen marknad. Det är mer relevant att se om processen kräver eller antar att det finns en kund. Kunden är då huvudintressent.

5.2.2 Kund och kontrakt

Det största problemet med RUP är dess starka koppling till en kund (*customer*) som agerar programvarubeställare för kontraktsdriven utveckling. Det är klart uttalat i ungefär hälften av de analyserade dokumenten och aktiviteterna, och underförstått i många andra. Kundcentreringen hade kunnat ses som ett rent formuleringsproblem, men allvarligare är att många dokument och aktiviteter görs enbart för att formalisera kommunikationen mellan kunden och utvecklande företag. Det som komplicerar bilden är att tre grupper av aktiviteter i RUP kan urskiljas. En grupp syftar till att kommunicera med en kund, en annan grupp nämner visserligen en kund eller kontrakt men aktiviteten kan utföras i vilket fall som helst och en tredje grupp varken nämner kund eller kontrakt eller antar att något sådant behöver finnas. Kund- och kontraktkopplingen är inte lika utbredd i alla arbetsflöden och därför är det intressant att titta på respektive arbetsflöde separat.

Project Management Workflow: Problemet med kund- och kontraktsorienteringen är komplicerat i *Project Management Workflow*. En del av dokumentationen och aktiviteterna är till för att åstadkomma insyn för kunden och att komma överens om arbetssätt:

> ”The Project Manager may not necessarily define the quality goals for the project, but ensures that these definitions are created and agreed by the customer, and captured ultimately in the Software Requirements Specification.” (beskrivning av aktiviteten Develop Quality Assurance Plan)

> "The Iteration Evaluation Criteria Review is a formal review of the tests and reviews that will be used to demonstrate to the customer that the objectives for an iteration have been met." (beskrivning av aktiviteten Iteration Evaluation Criteria Review)

Om vi bygger på resonemanget från den kvantifierade analysdelen, kan vi säga att dokumenten som hör till *Project Management Workflow* i något mindre grad är präglade av kontrakts- och kundorienteringen än dess aktiviteter. Egentligen är det bara *Product Acceptance Plan* som ofta är kopplat till att det finns en kund som accepterar den färdiga produkten. *Software Development Plan* (SDP) ger översikt över projektet och däri ingår många av de övriga artefakterna från både *Project Management Workflow* och *Requirements Workflow*. Innehållet i SDP ska gå att spåra till ett eventuellt kontrakt, men närvaron av ett kontrakt är inte nödvändig. *Measurement Plan* kan innehålla mätkrav från kontrakt, men inte nödvändigtvis. *Quality Assurance Plan* sammanställer kvalitetssäkrande procedurer från andra dokument och är till för att säkra en viss nivå av kvalitet för dokumentationen och för att säkra att kundens krav uppfylls. Det är kunden själv som ska godkänna kvalitetsmålen. Kvalitetsambitionerna kan finnas kvar även om det inte finns en beställare. Informationen kan samlas i ett policydokument som inte revideras inför varje utvecklingsprojekt. *Review Record* dokumenterar resultatet av ett granskningsmöte i största allmänhet behöver ingen kund.

Många av granskningarna som ligger under arbetsflödet *Project Management* har kunden som beslutsfattare. De omfattar fyra aktiviteter: *Iteration Evaluation Criteria Review, Iteration Acceptance Review, Project Acceptance Review* samt *Lifecycle Milestone Review*. Vid ytterligare en granskning bör kunden finnas representerad: *Iteration Plan Review*. De aktiviteter som hanterar produktacceptans har enbart relevans ur ett kontraktsperspektiv vilket medför att även *Develop Product Acceptance Plan* är beroende av organisationsform.

Därutöver finns det ett antal aktiviteter som i och för sig inte kräver varken kund eller kontrakt, men där det i RUP antas ändå att båda finns. Ett exempel är *Initiate Project*. En kund ska finnas med i planeringsgruppen, och acceptanskriterier för projektet ringas in. Å andra sidan kan man starta projekt utan en närvarande kund. Även *Plan Phases and Iterations* ska göras oberoende av kund, men här hamnar planen i SDP som sedan ska godkännas av kunden vid *Project Planning Review*. När en iteration ska värderas i *Assess Iteration* görs det mot kundens evalueringskriterier, men kontroll av diverse mätdata som inte behöver vara kundrelaterade görs också tillsammans med en omvärldsanalys, vilket är viktigt framför allt ur ett marknadsperspektiv. Samma sak gäller *Prepare for Phase Close-out* och *Prepare for Project Close-out*. Förväntningarna mellan teknikavdelningen och management ska stämmas av, vilket är extra viktigt om vid kontraktsbaserad utveckling, men inte helt meningslöst om det inte är det.

I övrigt handlar det om aktiviteter och dokument för att bestämma vad som ska mätas och därefter mäta enligt planen, identifiera risker och hur de ska hanteras, projektbemanning, göra en problemhanteringsplan och sedan agera enligt den, samt att övervaka och rapportera projektets status internt. Dessa aktiviteter kan göras oberoende av om syftet är marknadsdrivet eller kontraktsdrivet. Om allt dokumenteras blir omfånget kanske lite väl fylligt men varje organisation måste själv sätta gränsen för vad som är absolut nödvändigt.

Sammanfattningsvis kan man säga om *Project Management Workflow* att dokumenten i sig (undantaget *Product Acceptance Plan*) inte är beroende av att det finns en kund eller ett kontrakt, men att aktiviteterna däremot är präglade av det, vilket också framgår av tabell 5.1.1.1. Det stora antal kontrollaktiviteter och granskningar som utförs finns det troligen inte tid till i en marknadsdriven organisation. Däremot kanske det mer eller mindre formellt

tas någon form av motsvarande beslut. Antalet dokument som produceras måste relateras till projektets storlek och komplexitet, men det är svårt att ringa in vilka som helt borde utgå. Ett rimligt antagande är att en del dokument utformas som policydokument för företaget, vilket inte görs inför varje utvecklingsprojekt. Varje dokument måste kunna motiveras och det vore bra om det i RUP angivits vilka dokument som är mer viktiga än andra.

Business Modeling Workflow: Det minst komplicerade och mest konsekventa arbetsflödet är *Business Modeling Workflow* som genomgående antar att det finns en kund vars organisation ska modelleras för att sedan kunna göra ett system som fungerar i befintlig, eller omarbetad organisation. Det är också *Business Modeling Workflow* som ger de tydligaste riktlinjerna för hur hela arbetsflödet skulle kunna göras marknadsdrivet (se ovan) och säkert finns det mycket som kan vara till nytta vid modellering för en generell organisation på en marknad.

Requirements Workflow: Några av de mest centrala dokumenten i *Requirements Workflow* är uttalat producerade som kommunikationsmedel eller kontrakt med en kund:

> "The system analyst creates and maintains the Vision and Supplementary Specifications, which serve as input to the SRS and are the communication medium between the system analyst, the customer, and other developers. " (purpose for Software Requirements Specification)
>
> "A use-case model is a model of the system's intended functions and its surroundings, and serves as a contract between the customer and the developers." (introduction to Use-Case Model)
>
> "The customer approves the use-case model. When you have that approval, you know the system is what the customer wants." (introduction to the artefact Use-Case Model)

Visiondokumentet utgör ibland kontraktsbasen för relationen med kunden, men å andra sidan fyller det funktionen att sammanställa de tekniska kraven på hög nivå, vilket även kan vara nödvändigt i en marknadsdriven miljö, liksom att detaljerat beskriva kraven i artefakterna *Software Requirements Specification* (SRS). Detsamma gäller för *Supplementary Specifications*. I det andra exemplet från användningsfallsmodellen antas det till och med ett kontrakt, men syftet med dokumentet är inte enbart att fungera som kommunikationsmedel med kunden. Det fungerar också som kommunikationskanal mellan projektledning, utvecklare och testare inom programvaruföretaget. Eftersom dokumenten fyller även interna funktioner har de ett existensberättigande i en marknadsdriven organisation. Det viktigare av dessa tre dokument är *Vision*, medan SRS och *Supplementary Specifications* inte alltid görs i en marknadsdriven organisation. Principiellt sett påverkar inte kontraktsinriktningen arbetsflödet överhuvudtaget.

Ett problem som är besläktat med att det finns en kund är vad det programvaruutvecklande företaget har tänkt åstadkomma för kunden. Kunden har alltid ett "problem" som ska lösas, och i RUP uppmuntras till att leta efter "*the problem behind the problem*". I *Vision* anger avsnittet *Problem Statement* vad som ska åstadkommas för kunden och nyckelproblemen ska listas under avsnittet som behandlar "*Key Stakeholder or User Needs*". En *Stakeholder Request* utgår från att det finns ett problem som ska uppskattas och lösas. Det utesluter skapandet av nya behov inom företaget, eller sådana behov som är svåra för en intressent att sätta fingret på vid en intervju. Det är också svårt att fånga in till exempel utveckling av produkter kopplade till underhållningsbranschen där användaren kanske inte upplever sig ha större problem än brist på tidsfördriv. Tyvärr är problemfokuseringen mycket utbredd i RUP. Troligen går det att komma åt problemet relativt enkelt genom att skifta

tyngdpunkten till att efterfråga behov och lämna utrymme för skapandet av behov. Något enstaka dokument i RUP gör faktiskt det: i *Requirements Management Plan* utesluts inte marknadsdriven utveckling och där talar man om behov (*needs*) istället för problem.

5.2.3 Användare

Andra intressenter i RUP är användarna, vars önskemål ska identifieras under kravinsamlingen. Det är viktigt att förstå användarmiljön, oavsett om utvecklingen är kontraktsbaserad eller marknadsdriven. Användarna av produkten är väsentliga i båda utvecklingssätten, skillnaden är hur lättidentifierbara de är och hur delaktiga de är tänkta att vara i projektet. Vid användning av RUP ska utvecklingsgruppen känna till användarprofilerna, t ex vilket ansvarsområde som en användargrupp har. Det finns många aktiviteter där användaren är aktiv i RUP. Främst är det i samtliga aktiviteter till *Business Modeling Workflow* som användarna tas till vara:

> "… it's very important to involve those people who will work at this task in the new organization." (Purpose till arbetsflödesdetaljen Assess Business Status)

Såväl kund som slutanvändare finns med som stödjande roller, även om de bägge går under samlingsnamnet intressenter (*stakeholder*). Att modellera ett system utifrån en specifik organisation utan användarinblandning är fullt möjligt, men resultatet blir troligen inte uppskattat när det sätts i drift. En generell organisation skulle säkert kunna modelleras utan inblandning från användare eftersom systemet inte är beställt för att fungera på ett visst sätt. Däremot blir den generella modellen bättre ju mer kunskap som finns om olika organisationers arbetssätt och följaktligen användares förväntningar av ett programvarusystem.

Användarna tas också till vara i *Requirements Workflow*, när användningsfallen hittas och struktureras. Varje artefakt *Use Case,* som tillsammans utgör *Use-Case Model,* ska utformas så att både kunden och användarna förstår hur systemet är tänkt att fungera. Aktiviteterna som inbegriper detta är följaktligen *Detail a Use Case* och *Structure the Use-Case Model.* Att användare ska kunna förstå användningsfallen innebär att de är med i leken på ett tidigt stadium i utvecklingsfasen. Om det finns en beställare fungerar ju användningsfallsmodellen som ett kontrakt mellan kund och utvecklare och det är betydligt viktigare att på ett tidigt stadium kontrollera så att rätt produkt utvecklas. I det fallet finns det ofta användare tillgängliga som de facto kommer att använda systemet och skulle kunna avgöra om arbetssätten verkar rimliga. Möjligen är det i ett marknadsutvecklingsperspektiv inte så realistiskt att visa upp annat än prototyper som är mer eller mindre körbara och de användare som står till buds är inte alltid användare som senare kommer att använda systemet. Därmed inte sagt att användningsfallsmodellen står och faller med existens av identifierbara användare i ett beställarperspektiv, eftersom den fyller kommunikationssyften mellan programvaruarkitekter, designers, testare, projektledning, dokumentationsförfattare m fl.

5.2.4 Validering och speciella kravhanteringsaktiviteter

Analysverktyget tar fasta på enskildheter i dokument och arbetsflöden, men frågor som följer många steg genom RUP blir tyvärr för stora. Till exempel är validering något som i kontraktsdriven utveckling ska vara en "pågående process" där kunden fortlöpande får ge

sitt godkännande. Det krävs alltså en viss omfattning av insyn för kunden för att man ska kunna tala om kontraktsbaserad validering.

Vid uppdragsutveckling arbetar man typiskt med kravinsamling, modellering, validering och konflikthantering, där kravinsamling främst görs i början av projektet. Eftersom modellering i RUP mest ingår i arbetsflödet *Analysis and Design* ligger detta utanför analysen.

I RUP dokumenteras initiala krav i *Stakeholder Requests* och under aktiviteten *Elicit Stakeholder Requests* vaskas ytterligare krav fram som dokumenteras på samma sätt. Denna aktivitet avslutas dock i och med att arbetsflödesdetaljen *Understand Stakeholder Needs* i arbetsflödet *Requirements* är avslutad. Eventuella kravändringar eller nya krav som uppstår under utvecklingstiden hanteras i aktiviteten *Manage Change Request,* vilket antyder att mängden nya krav som kan tillkomma är begränsad och att de är tänkta att hamna i den version som utvecklas för tillfället. En ström av inkommande krav med tillhörande prioritering och versionsplanering hanteras inte i modellen och därför kan kravinsamlingen anses ha kontraktsinriktade drag.

Att validera systemet som utvecklas handlar om att säkerställa att rätt produkt utvecklas. Vid ett utvecklingsuppdrag är det kunden som avgör det uttryckligen. I RUP är formella granskningar (*reviews*) en viktig del av valideringen. I de tre arbetsflödena *Requirement Workflows, Business Modeling Workflow* och *Project Management Workflow* finns det sammanlagt inte mindre än elva aktiviteteter som styr formella granskningar[28] och i åtta av dessa finns det en uttalad kund som är mer eller mindre central för granskningen. Aktiviteterna *Iteration Acceptance Review, Project Acceptance Review, Lifecycle Milestone Review,* och *Review Requirements* hanterar godkännande av kraven, iterationen, fasen eller produkten vilka främst är relevanta om det finns en specifik kund. Att rätt system utvecklats för en marknad kan inte helt säkert avgöras förrän försäljningssiffror kan presenteras, vilket i så fall skulle göra de strikt kundorienterade granskningsmötena överflödiga. Det leder till slutsatsen att valideringen har kontraktsinriktade drag i RUP.

För att stämma av kriterier och planer finns det ännu fler granskningar och till slut blir det många möten i ett projekt. Om varje möte är en formell granskning, krävs att material distribueras i förväg vilket innebär ett antal dagar inför varje granskning där arbetet riskerar att bromsas upp tills mötet gett sitt godkännande. I en marknadsdriven organisation där utvecklingstiden ska hållas så kort som möjligt för att få ut produkten på marknaden fungerar alla dessa granskningar och möten som bromsklossar. Säkerligen är vissa beslut nödvändiga och måste tas, men kanske under mindre formaliserade former. Så som RUP ser ut nu är många av granskningarna enbart till för att få klartecken från kunden om att utvecklingen av deras produkt går på rätt håll.

Vid kontraktsbaserad utveckling finns en potentiell konfliktrisk mellan beställarorganisation och utvecklarorganisation som måste hanteras. Konflikterna riskerar att bli allvarligare då det är två organisationer inblandade och bägge anser sig ha rätt. Ett antal dokument och aktiviteter i RUP har som syfte att åstadkomma gemensamma förväntningar och minska konfliktrisken. Redan då projektet startas upp i aktiviteten *Initiate Project*, specificeras *Project Acceptance Criteria* som är en uppräkning av allt som ska levereras till kund inklusive sådant som eventuell utbildning för kundens personal, installationsfrågor och support. Detta verkar vara tänkt att placeras i SDP, även om informationen inte kan återfinnas varken i

[28] *Project Approval Review, Project Planning Review, Itration Plan Review, Iteration Evaluation Criteria Review, Iteration Acceptance Review, Project Acceptance Review, Lifecycle Miestone Review, PRA Project Review, Review Requirements, Review the Business Object Model,* samt *Review the Business Use-Case Model.*

dokumentmallen för SDP eller någon annanstans. Dessutom skapas artefakten *Product Acceptance Plan* som anger efter vilka kriterier som artefakterna kommer att bedömas. Även den placeras i SDP. Därutöver upprättas efter en formell granskning *Iteration Evaluation Criteria* för att kunna värdera varje iteration på ett för utvecklarna förutsägbart sätt och en *Quality Assurance Plan* med kvalitetsmålen godkända av kunden. Eftersom dessa dokument har en klar inriktning mot att minska konfliktriskerna, något som är typiskt för kontraktsinriktad utveckling, har RUP också i det här avseendet kontraktsinriktad karaktär.

Speciella kravhanteringsaktiviteter som gäller för marknadsdriven utveckling, d v s releaseplanering som balansering mellan kostnad och prioritet, hantering av ständig ström av inkommande krav hanteras inte. Visionen för produkten, inklusive positionering och marknadsinriktning och featureinnehåll ligger under systemanalytikerns ansvar, vilket inte tyder på marknadsinriktning. Förutsättningen är nämligen att kraven kommer in främst i början av projektet. Prioritering och sortering av krav görs först på ett ganska sent stadium där mycket arbete redan lagts på dem. Prioriteringen görs dessutom inte på affärsmässiga grunder utan görs av systemarkitekten tillsammans med intressenterna. Krav som kommer in senare tas om hand som ändringsbegäran inom pågående projekt. De kan visserligen få ett attribut för vilken release de är avsedda för, men det finns inget stöd för urvalsprocessen. Ett sådant arbetssätt är inte rimligt i en marknadsdriven organisation. Marknadsstödet är därför en läpparnas bekännelse i dokumenten utan uppbackning bland aktiviteterna.

Sammantaget saknas det aktivitetsstöd för typiska marknadsdrivna aktiviteter, och det är många aktiviteter och dokument som har som syfte att stämma av med kunden. Validering och de speciella kravhanteringsaktiviteterna får därför sägas vara inriktade på kontraktsutveckling. Eftersom kunden agerar beslutsfattare handlar det särskilt om *Project Management Workflow*, och inte fullt lika mycket om *Requirements Workflow* och *Business Modeling Workflow*. Det finns inte noterat någonstans vilka av dessa aktiviteter och dokument som är mer centrala än andra, vilket gör det tungrott för en marknadsdriven organisation att utgå från det arbetsflödet vid marknadsdriven utveckling.

5.2.5 Iterativ process

RUP är per definition en iterativ process men sådana metoder är enligt analysverktyget mindre vanligt vid kontraktsdriven utveckling. Eftersom det anses fördelaktigt att arbeta iterativt oavsett om man jobbar mot ett kontrakt eller ej, finns det inte någon nackdel med RUP av den anledningen.

5.2.6 Livscykel och utvecklarnas anknytning till programvaran

En stor skillnad mellan marknads- och kontraktsdriven utveckling är hur kopplingen till programvaran ser ut. Om programvaran är en egen produkt med fortlöpande releaser och uppdateringar är relationen långvarig och organisationen har ett större ansvar för kodens integritet. Då bör det finnas en plan för hur kommande releaser ska se ut, marknadens reaktioner på produkten och hur inkommande krav på förändringar ska tas om hand. Om programvaran är beställd betraktas arbetet som avslutat när kontraktet är uppfyllt och produkten accepterad och levererad.

I *Project Management Workflow* talas det sällan om ”produkter” utan mer om ”projekt”. Det är projekt som ska inledas i aktiviteten *Initiate Project*, planeras i *Plan Phases and Iterations,* och avslutas med *Prepare Project Close Out* och *Develop Product Acceptance Plan.* Dessa aktiviteter utgår från att kund och kontrakt som ska uppfyllas existerar, även om utvecklingen av en produkt för marknaden också kan bedrivas i projektform. Utgångspunkten är att produkten ska överlämnas till en kund och att när kontraktet är uppfyllt är arbetet avslutat. Åtagandet för programvaran är därför kortsiktigt. Däremot i den allmänt mer marknadsinriktade aktiviteten *Develop Business Case* finns möjligheten att produkten skulle kunna vara en vidareutveckling av en redan existerande produkt hos företaget. De ovan nämnda aktiviteterna är inte inaktuella som sådana i det fallet, men formuleringarna, utgångspunkterna och de vägledande frågorna som ställs blir missledande.

Artefakten *Stakeholder Request*, som ska ta hand om krav som kommer från kund, användare, marknadspersonal eller liknande, ger ett splittrat intrycket. Marknadspersonal kan framföra önskemål, vilket tyder på möjlighet till marknadsinriktad löpande kravinsamling. Där rekommenderas, att även om krav i första hand samlas in i under förberedelse- och etableringsfasen, ska krav tas in under hela projektets gång för förbättringar och uppdateringar till slutprodukten[29]. Åtagandet är alltså slut i och med att projektet är avslutat. Som källor till kraven anges inget som har med löpande kundkontakt via sälj-, support- eller marknadsavdelningar att göra, utan den enda kopplingen till något som skulle kunna liknas vid externa processer som marknadsundersökningar är formella intressentintervjuer. Problemet med dessa är att de är utformade för att undersöka en viss specifik organisation med problem som ska lösas, att de görs på vitt skilda intressenter fast med samma frågeuppsättning och att det är svårt att med denna typ av underlag undersöka marknaden för helt nya typer av produkter. RUP anger dock att intervjuerna ska modifieras efter behov. Grundproblemet kvarstår dock, RUP är inriktat på att lösa en organisations problem under ett projekts tak. Kanaler och aktiviteter för att ta in krav och att ta hand om dem under längre tid än under ett projekt är ett område som inte nämns.

Till RUPs fördel finns det rekommenderat i riktlinjerna för artefakten *Requirements Attributes* att varje krav ska ha planerad release som attribut, vilket antyder ett mer långsiktigt ansvarstagande för produkten. I malldokumentet till artefakten *Requirements Management Plan* står det att till varje *traceability item*, som är t ex en *Stakeholder Request*, ska det anges vilken release som en eventuell feature ska planeras till. Nackdelen är dock avsaknaden av aktiviteter som tar hänsyn till releaseplanering.

5.2.7 Systemmodellering

När det handlar om kravhantering i allmänhet, som i *Requirements Workflow*, är det specifika systemberoendet tydligast vid insamlingen av de initiala kraven. Utgångspunkten för artefakten *Stakeholder Requests* är att få fram kraven för systemet och däri kan ingå att få fram vilka externa källor som det nya systemet måste vara kompatibelt med. Denna kunskap är naturligtvis viktig att samla in också vid marknadsinriktad utveckling. I *Software Requirements Specification* ska det som är ”givet” hos kunden anges i ett särskilt kapitel, men det går inte att säga att SRS står och faller med detta.

Det arbetsflöde som är absolut tydligast i sina intentioner i fråga om systemmodellering är *Business Modeling Workflow*. Utgångspunkten är modelleringen av en specifik organisation,

[29] En kontroll i konceptet *Change Request Management* styrker att inkommande krav inte kan förläggas till annat än pågående release.

även om det som nämnts tidigare i kapitel 5.1.2 är möjligt att modellera en generell organisation med arbetsflödet. Den information som samlas in om målorganisationens system handlar inte bara om hårda fakta om processtrukturer, tekniska trender, verktyg, kunder och personalens kompetens, utan också om mjuka faktorer som attityder till förändring och om det finns så kallade nyckelpersoner som är mot förändring men som har inflytande över processförändringar. Denna inledande information samlas främst in under aktiviteten *Assess Target Organization*, vilket endast är aktuellt om det faktiskt finns en målorganisation[30]. Om utvecklingen gäller konceptuell modellering är det nödvändigt med kunskap om befintliga organisationer på den avsedda marknaden, vilken flora av organisationsstrukturer och processer som finns och vilka verktyg och applikationer som det måste finnas gränssnitt till.

Beslut fattas också om hur stor del av verksamheten som ska ingå i modellen och vilka gränssnitt mot omvärlden som måste finnas[31]. Dessa gränssnitt är här naturligtvis direkt kopplade till en specifik målorganisations behov, men skulle lika gärna kunna gälla generella behov som en organisation skulle kunna behöva. Aktiviteten handlar också huvudsakligen om att skapa nya strukturer och processer, vilket är aktuellt om en generell organisationsstruktur för en större marknad ska dras upp. Visionen för hur den nya organisationen ska tjäna på strukturen dokumenteras i artefakten *Business Vision*, vars malldokument tyvärr lider av samma begreppsförvirring som malldokumentet för *Vision* gör. Nyckelstrukturerna definieras sedan först i artefakten *Business Architecture Document,* och därefter utformas processerna på detaljnivå i artefakten *Business Use-Case Model* och *Business Object Model* med verksamhetsaktörer och användningsfall. Hela användningsfallsmodellen med ingående artefakter som *Business Use Case, Business Actor, Business Entity, Organization Unit, Business Worker* nämner den specifika organisationen som den kontext som alla delar ska fungera i, men samtliga dessa artefakter är aktuella även vid en konceptuell verksamhetsmodellering.

Egentligen är det bara den inledande aktiviteten *Assess Target Organization* som är intimt förknippad med en viss organisation. I övrigt finns det i och för sig tal om mer kulturella och kontextuella faktorer men det centrala som faktiskt görs behöver göras i alla fall, även om syftet är att åstadkomma ett generellt system. Det som saknas är i sådana fall aktivitetsstöd för att undersöka befintliga organisationer och att arkitektur, organisation och gränssnitt görs mer anpassningsbar för att maximera marknaden och göra införandet av processerna smidigare.

[30] Informationen samlas i artefakten *Target organization Assessment.*
[31] Görs i aktiviteten *Set and Adjust Goals.*

5.3 Övergripande analysresultat

Rational Unified Process

- är kontraktsinriktad.
- är problembaserad istället för behovsbaserad.
- hanterar endast ett utvecklingsprojekt åt gången.
- har endast stöd för marknadsdrivet tänkande i ett par dokument, som ej följts upp med genomgripande aktivitetsstöd.
- har många aktiviteter som går att göra även utan kund, men som ej behöver vara så formella.
- har många aktiviteter i arbetsflödet Project Management som hanterar överenskommelse med kund (granskningar, utformandet av utvärderingskriterier).
- saknar stöd för marknadsanalys.
- saknar hantering av ständigt inkommande krav, inklusive releaseplanering.
- saknar en löpande kontroll av marknaden och konkurrenterna med beredskap för att förändra produktinnehållet.
- saknar en roll som kan ta ett affärsmässigt ansvar för produkten.

Kapitel 6

Förslag till marknadsdriven RUP (MRUP)

6.1 Utgångspunkter och avgränsningar

De typiska aktiviteterna som stöder releaseplanering saknas i RUP och därför har ett nytt arbetsflöde *Release Planning Workflow* skapats. Det ska ta hand om de aktiviteter som kan påverka produktinnehållet, inklusive marknadsorienterade aktiviteter, att löpande samla in krav och att planera releaser. Arbetsflödet stöder produktens hela livscykel och sträcker sig alltså längre livscykelmässigt än *Project Manager Workflow*. Det medför konsekvenser för vad som hanteras av *Project Management Workflow* och *Requirements Workflow*. Vi har valt att inte försöka banta ner omfånget på Project Management, även om vi framfört mycket kritik mot antalet dokument som behöver framställas. Med en ”maximal” variant riskerar vi inte att utan tester eliminera dokument som upplevs som centrala av vissa organisationer. Enligt författaren till artikeln *Making RUP Agile* (Hirsch, 2002) är det fullt möjligt att skala ner RUP till att fungera väl även i mindre projekt. Det handlar om att välja lämpliga delar för projektet.

Ansvaret för produkten och dess innehåll är en ny roll, *Product Manager,* som delvis arbetar parallellt med *Project Manager*. En annan ny roll är *Requirements Database Management* som tar hand om inkommande krav och uppdaterar kravdatabasen.

Nu är en del av kravhanteringen flyttad till *Release Planning* vilket inte är helt självklart. Anledningen är att vi sett en poäng i att skilja ut den dynamiska verksamheten med hanteringen av inkommande krav och urvalsprocessen inför en release. Tillvaron för utvecklingsteamet kan hållas mer stabil genom att låta ändringar i pågående release gå via *Change Requests*. Det blir också tydligt att det är *Product Manager* som har ansvaret för innehållet i produkten, medan *Project Manager* ansvarar för utvecklingsarbetet.

Utanför modellen hamnar all hantering av produkten när den nått marknaden, marknadsmix och marknadsföringsarbete, PR, branding, tillverkning, logistik, personalfrågor, distribution, försäljning. Tyvärr är många förklarande texter till RUPs aktiviteter och dokument genomsyrade av kontraktsbaserade formuleringar och behöver genomarbetas för konsekvent genomförande.

6.2 Förslagets framväxt

Den feedback som kommit via fokusgruppmötena var oerhört givande. Det första förslaget som togs fram innebar en utvidgning av *Project Management Workflow* som också inkluderade produktledarens arbetsuppgifter. Däri ingick även andra uppgifter som produktledaren hanterar, som branding och att se till att göra användarens totala intryck av produkten komplett. Därefter flyttades fokus från att täcka in produktledarens samtliga arbetsuppgifter till att utveckla ett helt nytt arbetsflöde: *Release Planning*. Detta skulle fortfarande täcka in produktledarens ansvarsområden, men enbart hantera de som direkt påverkar produktinnehållet. Det betyder också att aktiviteter som hanterar den färdiga produkten utgick. Under den första utvärderingen framkom också att värdeanalys och releaseplanering utförs växelvis tills releaseplanen är tillfredsställande. I det stora hela var grundstrukturen därefter färdig.

Under den andra utvärderingsomgången blev det uppenbart att gränsen mellan arbete som görs för en release och arbete som utförs löpande för hela produkten måste göras tydlig. Likaså gränsdragningen mellan det strategiska arbetet och utvecklingsarbetet. Informationsutbytet mellan marknads- och utvecklingsavdelningen inför releaselanseringar var något som upplevts som mycket uppskattat i deras respektive organisationer och lades till i modellen. Andra insikter som nåddes under iterationen gällde tidsuppskattningar i tre nivåer, interna värdeanalyser , viktiga beslutspunkter och projektintegrering.

Under tredje iterationen gjordes aktiviteterna för gallring, kostnadsuppskattning och beroendeanalys tydligare. En livscykelmodell för krav infördes för att visa på hur informationen om krav växer fram.

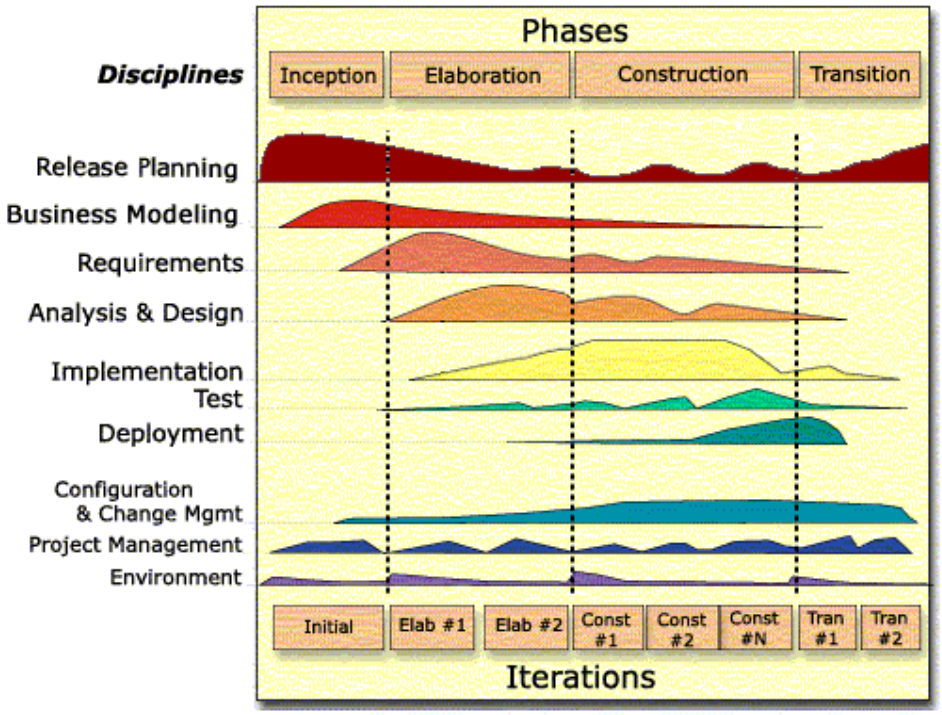
Under den fjärde iterationen förlades gränssnittet mellan det strategiska arbetet och utvecklingen till en ursprunglig projektplan i SDP. Det framkom också att våra ambitioner att banta ner arbetsflödet för projektledning kunde bli svårt eftersom skillnaderna i vilka dokument som i realiteten används är stor mellan olika typer av organisationer. Vi valde också att kalla beslutspunkterna för granskningar, eftersom organisationen i ett *Development Case* ska specificera i vilken form granskningarna ska genomföras. Därför togs beslutet att behålla det arbetsflödet i en ”maximal” version.

Under den sista iterationen framkom egentligen inget nytt. Den fungerade mer som en kontrollpunkt att industrin kunde känna igen sig i MRUP. En kommentar gällde omfånget på *Project Management*, som ansågs lite väl stort. Däremot verkade modellen rimlig för informanterna och att arbetssättet nog skulle fungera, om det först testades i skarpa projekt. Överhuvudtaget har mottagandet varit positivt bland informanterna.

6.3 Förslag till MRUP

6.2.1 MRUPs processtruktur

Eftersom ett nytt arbetsflöde införts har processtrukturen modifierats något (jämför figur 3.5.4). I figur 6.2.1.1 är en skiss på hur strukturerna förändrats.



Figur 6.2.1.1 *Processtrukturen för MRUP.*

Release Planning Workflow är aktivt under alla faserna. Det täcker produktens livscykel medan de övriga är projektinriktade. Eftersom en del av det arbete som utfördes av *Requirements Workflow* och *Project Management* nu utförs av *Release Planning*, har det inneburit en viss förskjutning för när de påbörjas.

6.2.2 Nya eller modifierade roller

6.2.2.1 Product Manager

Product Manager (produktledare) är en ny roll som har ansvar för produktens innehåll. Denna roll kan se mycket olika ut på olika företag. Här ser vi rollen som att den dels är ansvarig för produkten, och dels representerar marknadsavdelningen. På de ställen i modellen där produktledaren har ansvaret finns ofta också projektledaren med som stödjande roll, och vice versa med projektledaren när beslut som rör hela produkten ska tas. Det är nödvändigt att ha en särskild roll som har ansvar för hela releasen som ju kan bestå av många delprojekt som drivs av sin respektive projektledare.

Produktledaren har sitt arbete koncentrerat till det nya arbetsflödet *Release Planning* och ansvarar för att utarbeta *Vision*- och *Business Case*-dokument, samt risklista och riskhanteringsplan. Allt som har med marknadsrelaterade undersökningar är produktledarens ansvar, liksom att värdeanalysen genomförs och att planera releasen. En annan viktig uppgift är att utse någon som hanterar kravdatabasen och att dra upp riktlinjer för hur kravinsamlingen ska gå till och vilka kundinterface som måste tas om hand. Observera att en produktledare i allmänhet har långt fler arbetsuppgifter än de som beskrivs i MRUP. Här har enbart de som påverkar produktinnehållet, d v s programvaru-utvecklingen tagits med.

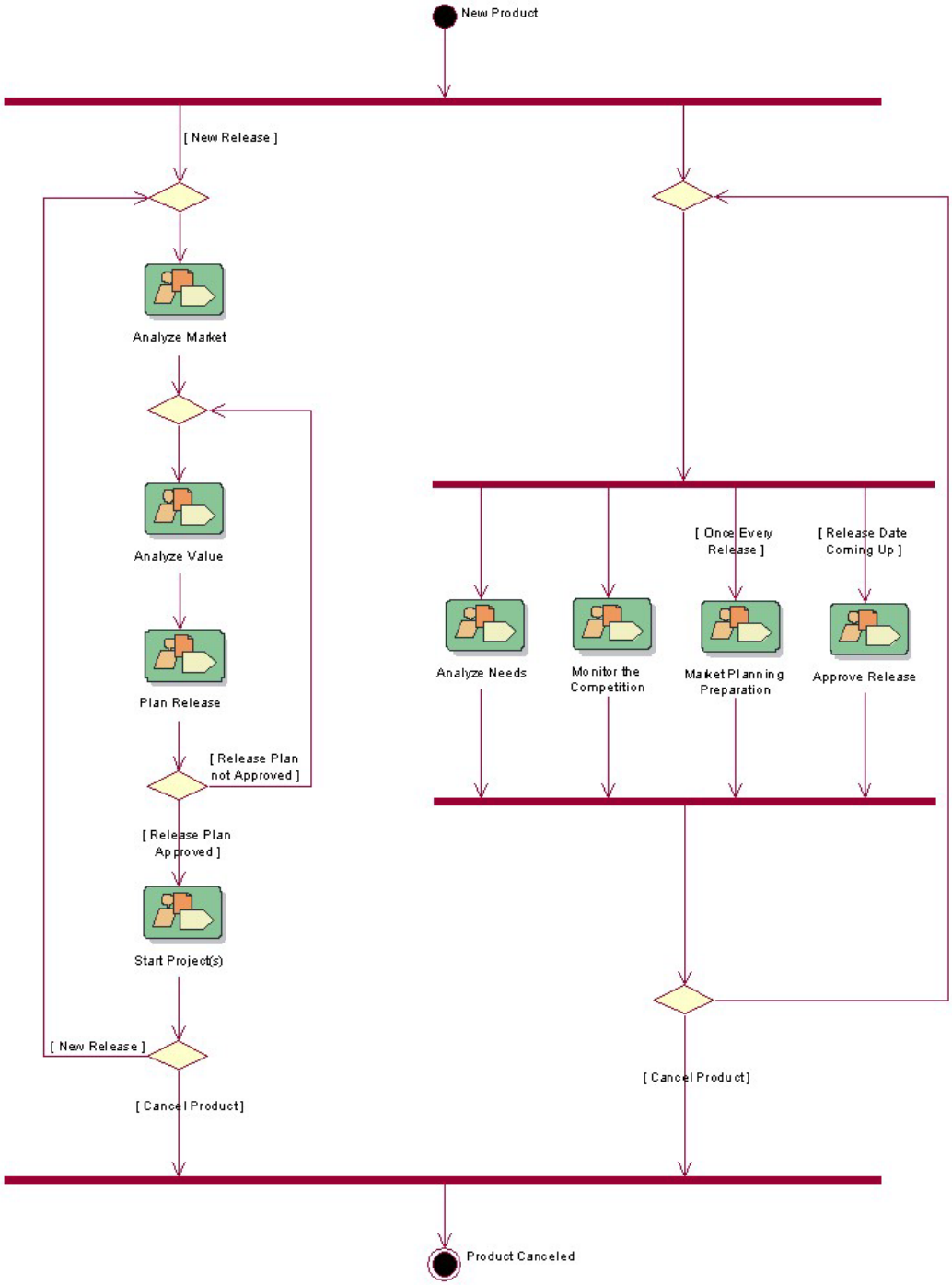
6.2.2.2 Requirements Database Management

En flaskhals för marknadsdrivna programvaruföretag är hantering av inkommande krav. Att verkligen prioritera arbetet och att tillsätta särskild personal är som sagt viktigt. En ny roll är därför *Requirements Database Management*. Vid hantering av inkommande krav sorteras dubbletter och de krav som kan ses som ”oviktiga” bort vid en första *screening*. Vid uppdatering av statusattributen av kraven till *applied* bör också en utgallring göras för att inte lämna kvar krav som är inaktuella p g a implementerade krav.

6.2.2.3 Stakeholder

Stakeholders, eller intressenter, är i RUP alla som har ett materiellt intresse av hur slutprodukten blir. Det betyder ofta slutanvändare eller slutkund, men också marknadspersonal och utvecklare i största allmänhet. Just marknadspersonalen är några som i ursprungliga RUP inte tas till vara som intressenter, men är viktiga i MRUP.

6.2.3 Release Planning Workflow

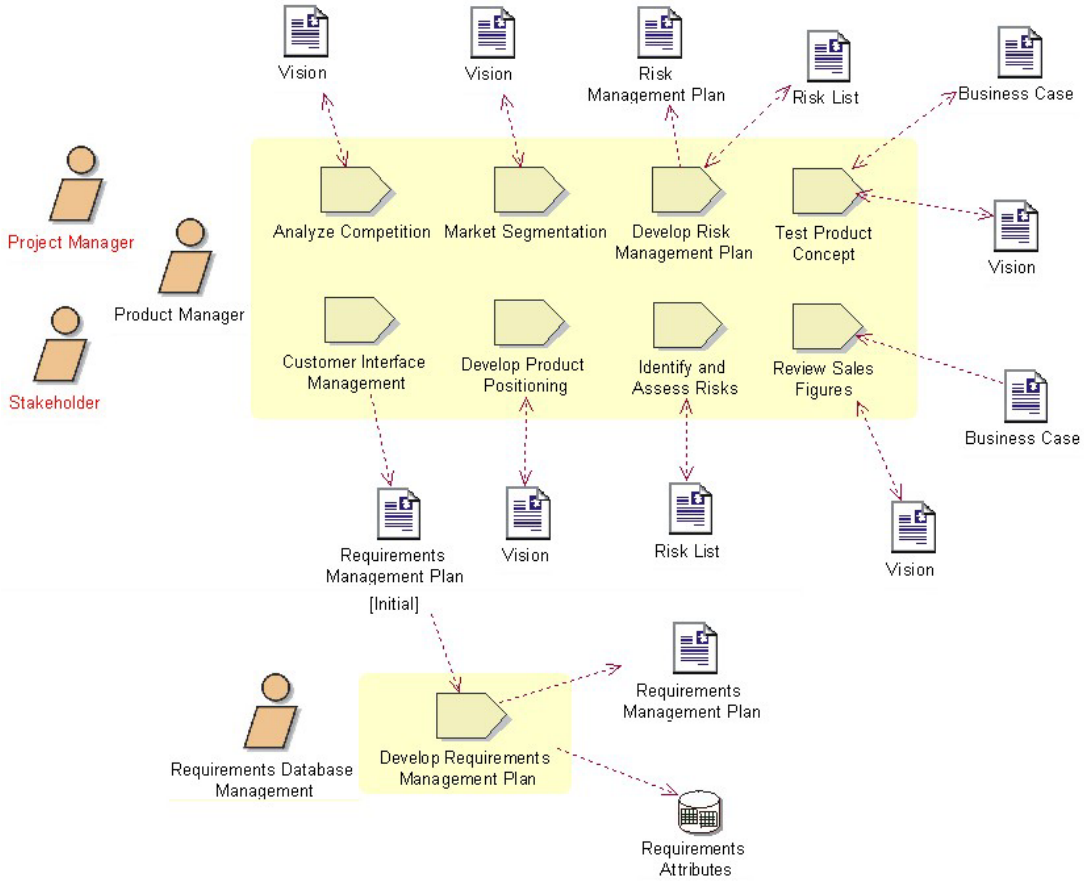


Figur 6.2.3.1 *Release Planning Workflow*

Detta nya arbetsflöde stödjer dels den löpande planeringen av releaser, dels det löpande arbetet med kravinsamling och övervakning av konkurrenterna under hela produktens livscykel. Resterande arbete som hanterar produkten, men som inte påverkar produkt*innehållet*, finns inte med i modellen. Nyckelrollerna innehas av *Product Manager* och *Project Manager* som beroende på arbetsfördelningen inom företaget tar hand om många av

aktiviteterna. Ett nytt *Vision*-dokument har utformats där all information om produkten, releasen och dess marknad finns sammanställd. Också ett omformulerat *Business Case* finns för att samla den affärsrelaterade informationen. De nya dokumenten finns i Appendix A. Ännu en ny roll, *Requirements Database Management*, samlar in och administrerar nya krav. Observera i beskrivningarna att de aktiviteter som är direkt hämtade från RUP inte motiveras i detalj, däremot anges att källan är RUP.

6.2.3.1 *Analyze Market*



Figur 6.2.3.1.1 *Analyze Market*

I bildmaterialet kommer den roll som är huvudansvarig att stå närmast fältet med aktiviteterna. I det övre fältet i figur 6.2.3.1.1 är alltså *Product Manager* huvudansvarig och *Project Manager* och *Stakeholder* är stödjande roller.

Releasearbetet bör inledas med en grundläggande marknadsanalys. Resultatet kommer att påverka produktens innehåll och utgöra beslutsunderlag inför releaseplaneringsarbetet. En pedagogisk poäng är att synliggöra detta arbete för utvecklarna. Den stödjande intressentrollen är här mest marknadspersonal och kunder. Notera att portföljanalyser inte hanteras av RUP. Se i övrigt kapitel 3.4.5.

- Analyze the Competition:
  Konkurrenternas agerande är till viss del avgörande för vilken produkt som företaget väljer att utveckla.
  - Kartlägg konkurrenterna.

  - Identifiera konkurrenternas mål och strategier (kan vara svårt).
  - Analysera konkurrenternas styrkor och svagheter.
  - Välj konkurrentstrategi (t ex kostnadsledande, differentiering, fokusering).

- Market Segmentation:
  En smart segmentering kan underlätta valet av produktinnehåll. Bör göras noggrant. Se kapitel 3.4.2 för närmare presentation.
  - Välj segmenteringsattribut.
  - Välj ut attraktiva segment med avseende på storlek, tillväxtmöjligheter, lönsamhetsnivåer, konkurrenssituation.

- Develop Risk Management Plan:
  Görs eventuellt inte inför varje projekt, utan kan ingå i företagets gemensamma rutiner. Källa: RUP.
  - Skapa en dokumenterad plan med procedurer för att identifiera, analyser och prioritera risker.
  - Skapa en risklista.

- Test Product Concept:
  Görs framför allt vid utveckling av helt ny produkt. Görs på valfritt sätt. Källa: utvärderingar.

- Customer Interface Management:
  Det är mycket viktigt att det finns någon som är speciellt ansvarig för kravdatabasen. Det är också en klar fördel att ha klart varifrån kraven kan komma och hur de ska samlas in. Se i övrigt kapitel 3.2.2.
  - Utforma en grupp som ansvarar för inkommande krav: Requirements Database Management. Lägg huvudansvaret på en specifik person.
  - Identifiera kundinterface, t ex user groups, support, säljavdelning, marknadsavdelning.
  - Utforma riktlinjer för att samla in kraven.
  - Om kravinsamlingen sker i en distribuerad miljö, fördela ansvaret för kravinsamlingen från lämpliga områden.

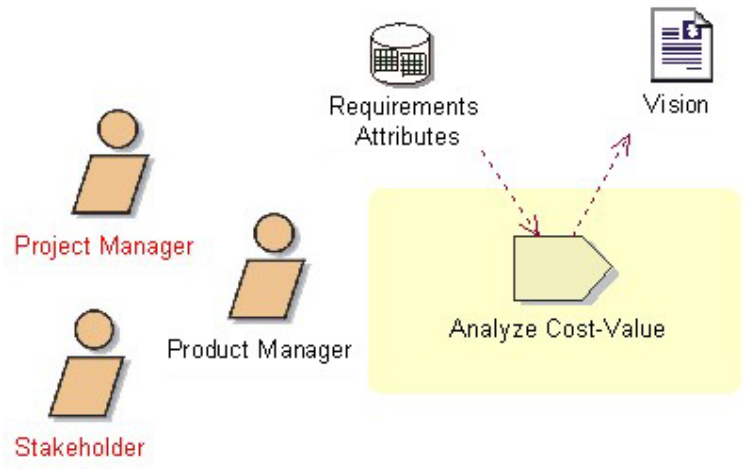
- Develop Product Positioning:
  Se kapitel 3.4.4 för närmare diskussion.
  - Kartlägg produktens nuvarande positionering inom respektive marknadssegment och relatera till relevanta konkurrenters.
  - Bestäm mål för produktens positionering inom respektive marknadssegment.

- Identify and Assess Risks:
  Källa: RUP
  - Identifiera potentiella risker (resursrisker, affärsrisker, tekniska risker, schemarisker)
  - Analysera och prioritera risker
  - Identifiera strategier för att undvika och begränsa risker

- Review Sales Figures:
  Om det gäller en existerande produkt är det viktigt att ta tillvara den statistik som finns. Vi har inte specificerat ett särskilt RUP-dokument för detta. Även annan statistik om produkten är intressant att titta på här. Källa: utvärderingar.
  - Titta på nuvarande siffror och bedöm utvecklingstrend under lämplig period. Analys av var i livscykeln produkten befinner sig.
  - Sales forecasting
  - Eventuella siffror om kundtillfredsställelse från t ex support- och serviceavdelningen.
  - Studera eventuella win-lossrapporter.

- Develop Requirements Management Plan
  Gör detaljarbetet som påbörjats i Customer Interface Management, men eventuellt inte för varje projekt. Finns i original-RUP, men har flyttats fram. Källa: RUP, men se också kapitel 3.2.1 och 3.2.2.
  - Utveckla en plan för hur kraven ska dokumenteras, deras attribut och riktlinjer för spårbarhet och hantering.
  - Koppla till verktyg
  - Dokumentera

6.2.3.2 Analyze Value



***Figur 6.2.3.2.1** Analyze Value*

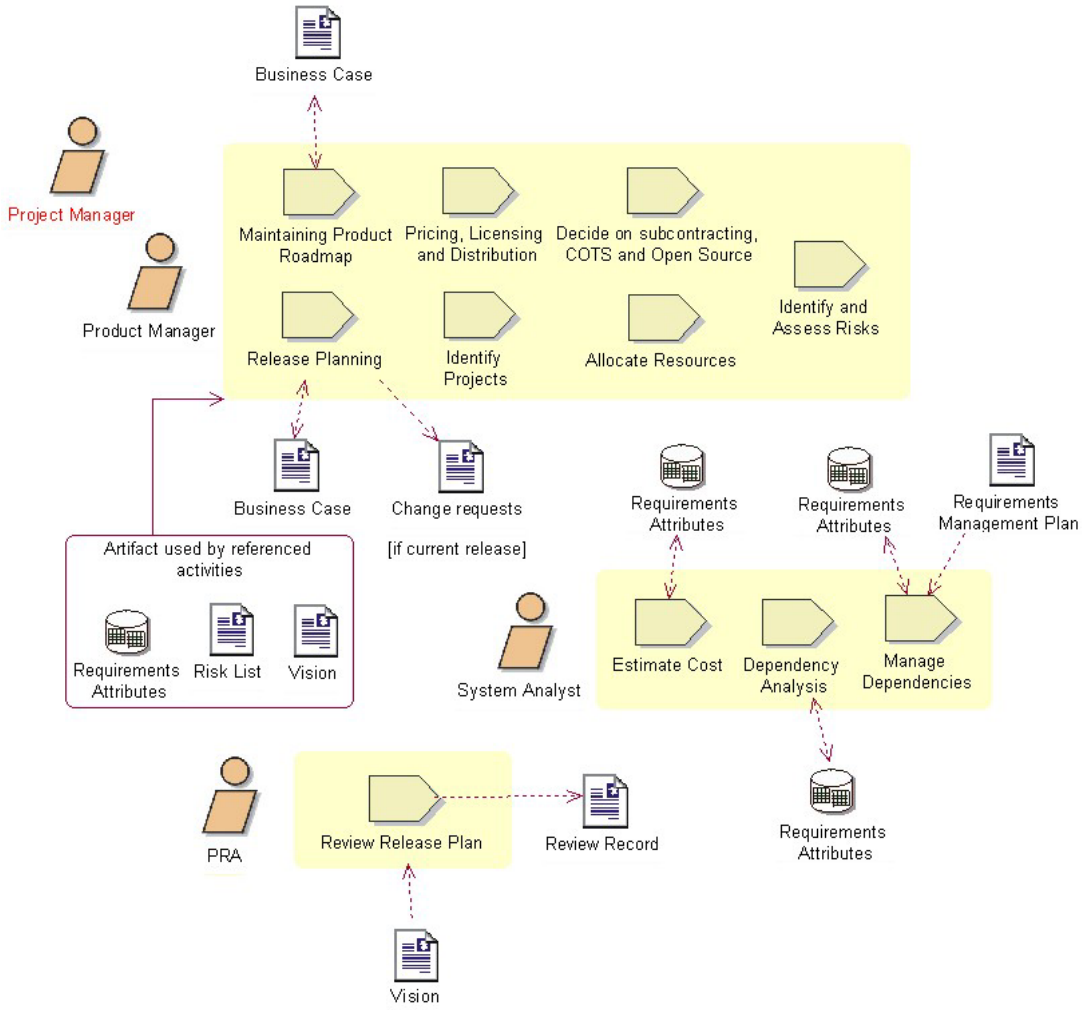
Resultatet av värdeanalysen används till releaseplaneringen och underlättar arbetet med att bestämma hur ”rätt” produkt ska utvecklas. Med värdeanalys avses en prioritering ur olika intressenters synvinkel m h a parvisa jämförelser. Resultatet blir då i stort sett en lista med features i prioriterad ordning, men vid användning av ett program som Focal Point kan resultatet visualiseras och olika intressentgruppers värdeanalyser smidigt jämföras. Det handlar alltså dels om kundvärdeanalys, och dels om intern värdeanalys av en viss feature (marknadsföringsvärde, ”teknisk höjd”, eller annat internt värde). En värdeanalys görs per marknadssegment, och en eller flera för det interna värdet. Glöm ej att kontrollera värde för konkurrenternas produktfeatures. Se kapitel 3.2.4.

- Analyze Cost-Value:
  För varje marknadssegment:

- Identifiera de krav du vill ha värderade
- Välj kundrepresentanter (kan vara reella kunder, eller bara representanter som hämtas internt från t ex sälj- eller marknadsavd).
- Gör en värdeanalys, t ex med Focal Point
- Visualisera

Gör det samma för den interna värdeanalysen, fast med representanter för t ex marknads-, sälj- och utvecklingsavdelning. Värdeanalysen görs med fördel i grupp för att stimulera diskussion om kraven.

6.2.3.3 Plan Release



Figur 6.2.3.3.1 *Plan Release*

Att planera en väl sammansatt release är komplicerat. Vilka avvägningar som är avgörande är specifikt för varje organisation vid olika releaser. Arbetsflödesdetaljen syftar till att stödja med aktiviteter för att åstadkomma ett bra underlag för att fatta beslutet. Exakt *hur* valet av ingående features ska göras går det inte att säga något om. Se i övrigt kapitel 3.2.6.

- Maintain Product Roadmap:

Det viktigaste är att fastställa releasedatum, vilket påverkar hur mycket features som kan inkluderas i releasen. Källor: Leffingwell, 2002 och utvärderingar.

- Uppdatera viktiga datum (release, externa events, viktiga milstolpar i utvecklingen).

- Pricing, Licensing and Distribution:
  Om särskilda hänsyn behöver tas vid utformningen av produkten, notera det som features i Vision. Källor: RUP; Leffingwell (2002).

- Decide on Subcontracting, COTS and Open Source:
  Det kan vara billigare att använda färdiga komponenter, eller så kan specialistkompetens användas för vissa hela delprojekt. Open Source kan också bli billigare att använda men innebär risker om resterande delar av moduler också blir att anse som Open Source. Källa: utvärderingar.
  - Hur mycket ska läggas ut på underleverantörer. Kostnadsuppskattning efter offerter.
  - Finns det lämpliga COTS och till vilken kostnad inklusive tidsuppskattning för integrering.
  - Om Open Source-kod kan integreras, finns det risk att resten av modulen blir Open Source?

- Release Planning:
  Att planera en release är mycket komplicerat, men ett bra underlag i fråga om kostnadsuppskattningar, beroenden, och värde för företag och marknadssegment är en bra hjälp på vägen. Denna aktivitet hanterar också enstaka features som ska planeras till pågående release. Utfärda då en Change Request som sedan hanteras av arbetsflödet Configuration and Change.
  - Givet releasedatum och kostnadsuppskattning, beroendeanalys och värdeanalys för varje krav, skapa en balanserad release som maximerar lönsamheten. Man kan planera flera releaser samtidigt.
  - Se till att värdet inom prioriterade segment blir tillfredsställande.
  - Se till att värdet för företaget (främst marknadsföringsvärdet) blir tillfredsställande.

- Identify Projects:
  Ett naturligt steg eftersom features tenderar att hänga ihop.
  - Dela in releaseplanen i lämpliga delprojekt.
  - Fördela features som är förlagda till senare releaser till delprojekt.
  - Notera i Vision.

- Allocate Resources:
  Handlar om organisationsövergripande beslut om vilka projekt som ska läggas på vilka team. Det är viktigt att samtidigt se över om det finns projekt som inte går framåt, som kanske kan läggas ner för att fokusera kraften på prioriterade projekt. Källor: utvärderingar, Carlshamre, 2001, Karlsson, 1998, Karlsson, 2002.
  - Se till att få resurser tilldelade till planerade projekt.
  - Lägg ner vilande projekt med mycket låg prioritet.
  - Revidera releaseplanen om ej resurser fås.
  - Tillsätt projektledare.

- Identify and Assess Risks:
  Källa: RUP
    - Identifiera potentiella risker (resursrisker, affärsrisker, tekniska risker, schemarisker)
    - Analysera och prioritera risker
    - Identifiera strategier för att undvika och begränsa risker

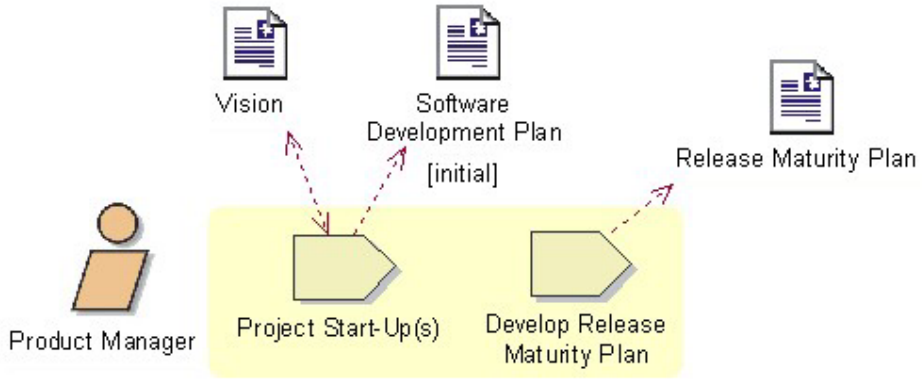
- Estimate Cost:
  Görs tre gånger: vid Plan Release, Analyze Needs och Manage the Scope of the System. Den första estimeringen är en mycket grov uppskattning, och mer detaljerad och noggrannare vid releaseplaneringen. Även tidsuppskattningar kostar pengar så ”lagom tid” måste avsättas på varje nivå. Slutligen vid Manage the Scope of the System görs den noggrannaste uppskattningen, eftersom den behövs vid iterationsplaneringen. Exakt vilka metoder som används avgörs av respektive organisation, men vi utgår från att en expert tar övergripande ansvar. Se i övrigt kapitel 3.2.3.
    - Bedömning av tidsåtgång för implementation av varje feature. Basera på uppskattning om hur många komponenter som påverkas. Gör mer omfattande förstudie om det är något som är komplicerat eller tar lång tid.
    - Uppskatta kostnader för andra resurser: licenser för ingående delar, hårdvara etc. Detta är dock en budgetfråga och ingår ej i värdeanalysen som är tidsbaserad.

- Dependency Analysis:
  Görs både vid Plan Release och Analyze Needs, men mer detaljerat vid releaseplaneringen. Beroenden är det som komplicerar releaseplaneringsarbetet mest. Tyvärr finns det inga färdiga metoder för att analysera och hantera beroenden. En grovuppskattning av beroenden på hög nivå bör dock vara möjlig att göra för någon med god insikt i produkten. Se i övrigt kapitel 3.2.5.
    - Undersök om det finns några beroenden med andra krav. Beroenden kan exempelvis vara av typen AND, REQUIRES, CVALUE, ICOST.
    - Uppdatera status till *Classified,* om det ännu är *Assigned* när både beroendeanalys och kostnadsuppskattningen gjorts.

- Manage Dependencies
  Källa: RUP.
    - Tilldela attribut, etablera och verifiera spårbarhet.
    - Hantera ändringar av krav.

- Review Release Plan:
  Innehållet i releaseplanen påverkar affärsmöjligheterna för företaget och bör beslutas av högre instans (PRA).
    - Godkänn releaseplanen.

6.2.3.4 Start Project(s)



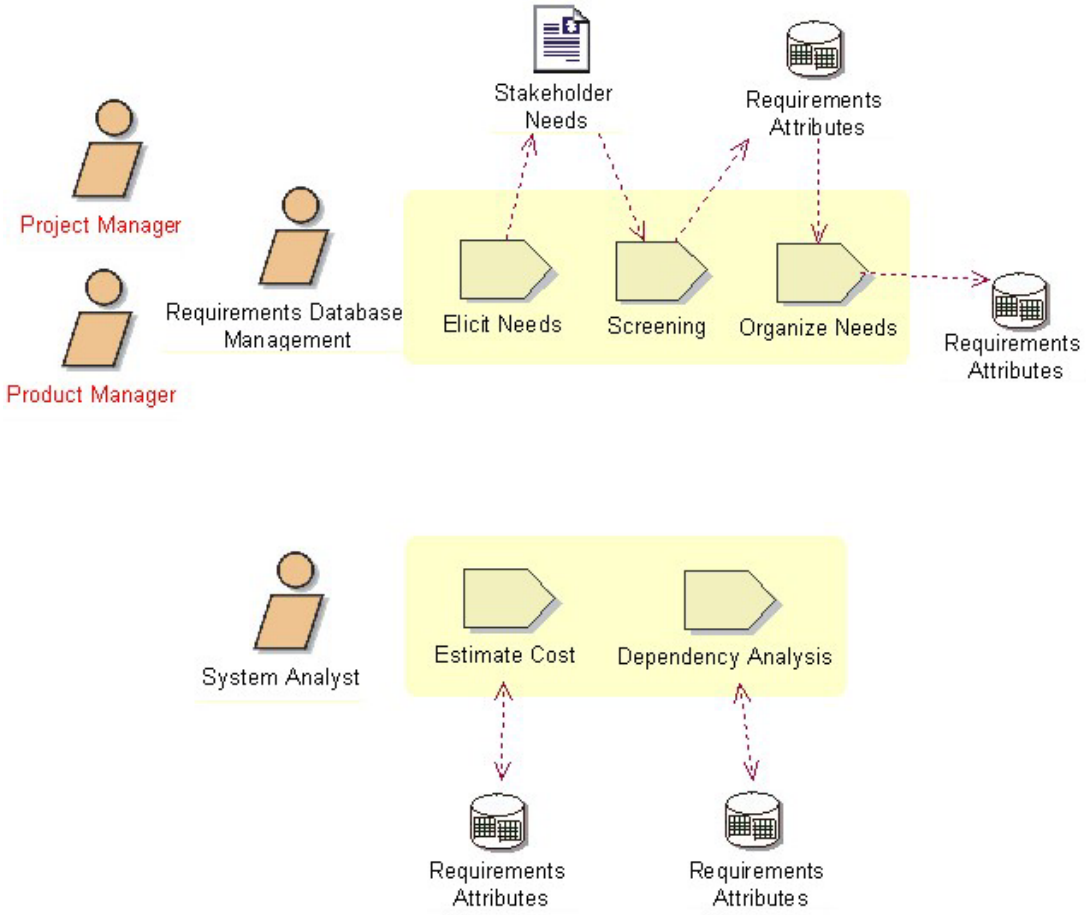
Figur 6.2.3.4.1 *Start Project(s)*

En naturlig följd av att projekt beslutats. Innebär att början till projektdokument (SDP) dras upp och fungerar som gränssnitt mellan det strategiska arbetet och utvecklingsarbetet.

- Develop Release Maturity Plan
  Specificera vad som ska vara klart till release, hur acceptanstestning ska gå till och vem som godkänner.

- Project Start-Up(s):
  Projekten som startas behöver inte nödvändigtvis vara de som ska vara med i nästkommande release. Det kan lika gärna handla om mer långsiktiga projekt som kommer att integreras längre fram.
    - Specificera övergripande projektbeskrivning, inklusive budget, datum för alpha- och betaversion i en första version av SDP.
    - Innebär att arbetsflödet Project Management börjar med sin första arbetsflödesdetalj: Conceive New Project. Starta samtliga projekt.

6.2.3.5 Analyze Needs



Figur 6.2.3.5.1 *Analyze Needs*

Arbetsflödesdetaljen styr det löpande arbete som ska göras när krav kommer in. Tillräckliga resurser bör tilldelas för att effektivisera det kommande releaseplaneringsarbetet.

- Elicit Needs:
  Detta ska göras löpande för produkten. Vid kravinsamling är det mycket viktigt att specificera det bakomliggande behovet (rationale). Det är också fullt möjligt att samla in behov och spara med ett attribut som anger att det är ett mål-krav. Se kapitel 3.2.1 och 3.2.2.
    - Samla in krav via de etablerade kanalerna
    - Samla in krav internt
    - Ta med krav som konkurrenter positionerat som nödvändiga.
    - Använd någon av metoderna i ACRE (se kap 3.2.2)
    - Kraven är beskrivna enligt specifikationerna från Customer Interface Management med t ex titel, beskrivning, rationale, källa, kundfokus och konkurrentfokus.

- Screening:
  Hanteringen av inkommande krav riskerar att bli en flaskhals om inte den ansvarige för kravdatabasen har viss disciplin med vad som tillåts läggas in. ”Oviktiga” krav bör därför sållas bort direkt. Det kan visserligen vara riskabelt att

kasta krav på en tidig nivå, men riskerna med överbelastning är mycket allvarligare för releaseplaneringen. Se kapitel 3.2.2.

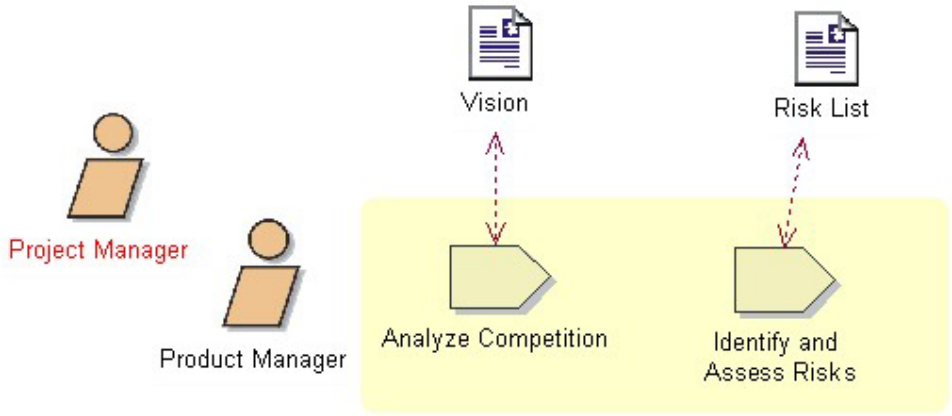
- Grovsålla kraven, d v s prioritera bort ”ointressanta” krav. Se också till att inga dubbletter läggs in.

- Organize Needs:
  Det administrativa arbetet med nya krav. Attributen är de som tidigare definierats vid aktiviteten Requirements Management Plan. De viktigaste i det här stadiet är att se till att kostnadsuppskattning och beroendeanalys görs. Källa: Karlsson, 2002 behovsanalys, s 43 ff.
  - Lägg in kravet i databasen och tilldela kravet till lämplig expert som sätter attribut för tidsuppskattning, påverkade moduler och beroenden. Ett bäst-före-datum kan också vara användbart.
  - När kravet är tilldelat till expert blir status *Assigned,* enligt livscykeln i figur 3.1.1.2.

- Dependency Analysis:
  Se kap 6.2.3.3.

- Estimate Cost:
  Se kap 6.2.3.3.

6.2.3.6 Monitor the Competition



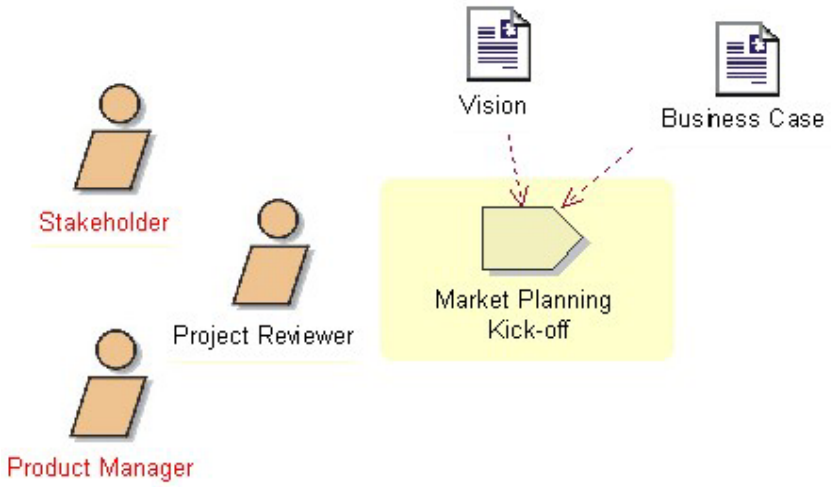
Figur 6.2.3.6.1 *Monitor the Competition*

Att titta på konkurrensen enbart när det är dags att planera en ny release är en smula enögt. Man bör ständigt hålla ett vakande öga på konkurrenternas agerande och produktreleaser för att se hur det påverkar den egna produktens konkurrenskraft, och eventuellt göra justeringar i releaseplanen.

- Analyze the Competition:
  Konkurrenternas agerande är till viss del avgörande för vilken produkt som företaget väljer att utveckla.
  - Kartlägg konkurrenterna.
  - Identifiera konkurrenternas mål och strategier (kan vara svårt).
  - Analysera konkurrenternas styrkor och svagheter.
  - Välj konkurrentstrategi (t ex kostnadsledande, differentiering, fokusering).

- Identify and Assess Risks
  Källa: RUP
    - Identifiera potentiella risker (resursrisker, affärsrisker, tekniska risker, schemarisker)
    - Analysera och prioritera risker
    - Identifiera strategier för att undvika och begränsa risker

6.2.3.7 Market Planning Preparation

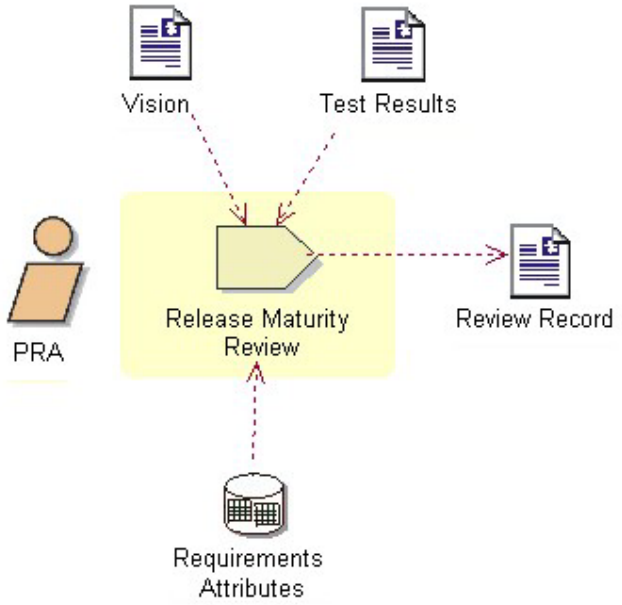


***Figur 6.2.3.7.1** Market Planning Preparation*

Arbetet med att planera marknadsföringen ligger utanför RUP men marknadsavdelningen behöver veta exakt vad som kommer att finnas i nästa release. Ett gemensamt möte med marknadsavdelningen och representanter från de olika projektgrupperna är ett bra sätt för marknadsavdelningen att både veta vad som ingår i releasen, och få en fördjupad förståelse av nya features för att kunna marknadsföra dem bättre. När detta möte infaller beror lite på hur långa releasecykler företaget använder, men förslagsvis någon gång mitt i releasearbetet (en gång per release). Källa: utvärderingar.

- Market Planning Kick-off:
  Genomgång av vad som verkligen kommer att ingå i nästa release. Stakeholders är representanter från marknadsavdelning och utvecklare från alla projekt som kommer integreras i nästa release.

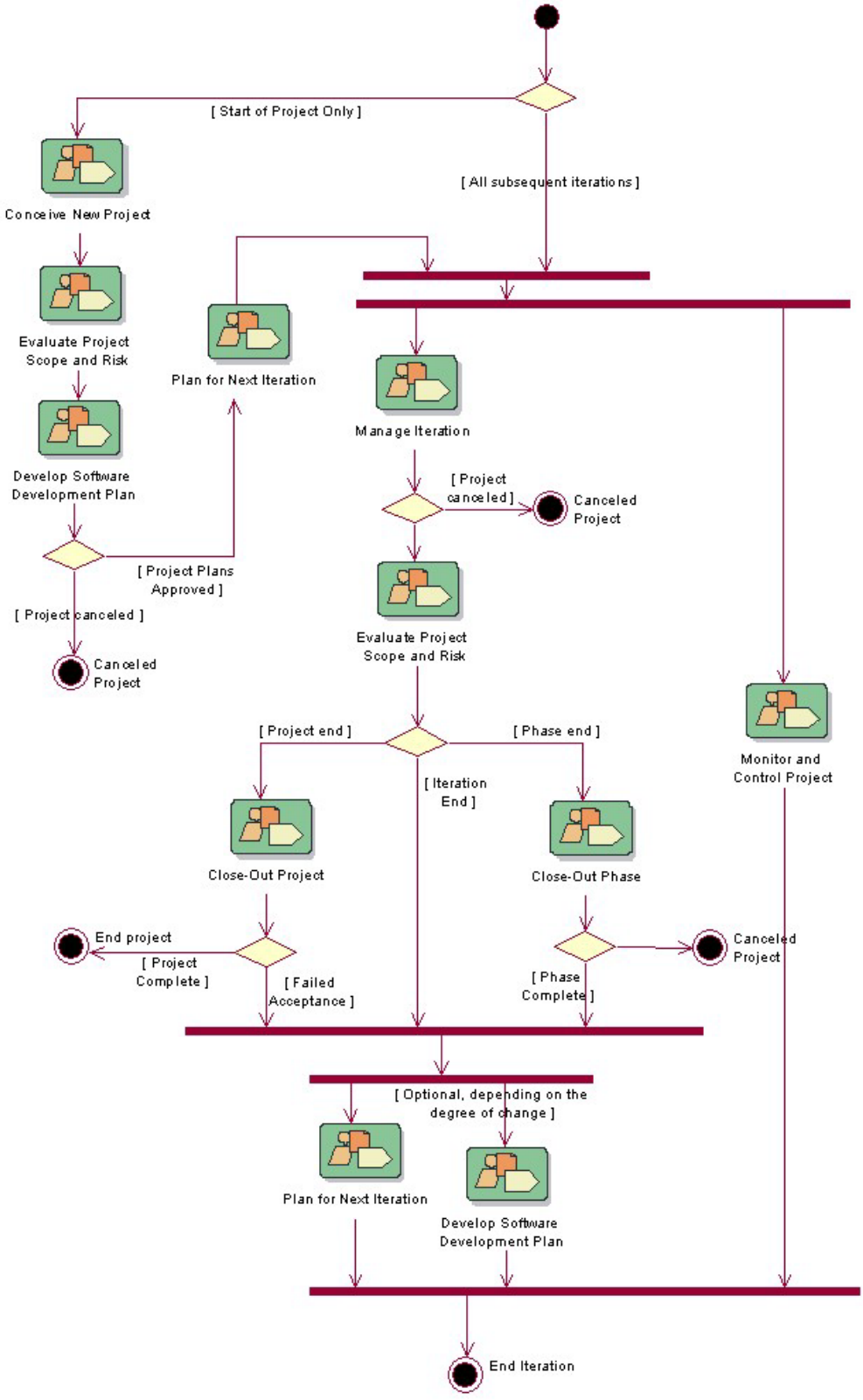
6.2.3.8 Approve Release



Figur 6.2.3.8.1 *Approve Release*

- Release Maturity Review
  En ny produkt bör inte släppas lättvindigt. PRA godkänner att produkten nu är klar, enligt de kriterier som ställt upp. Här bör kontrolleras att alla features blivit implementerade och att releasen håller tillräckligt hög kvalitet.

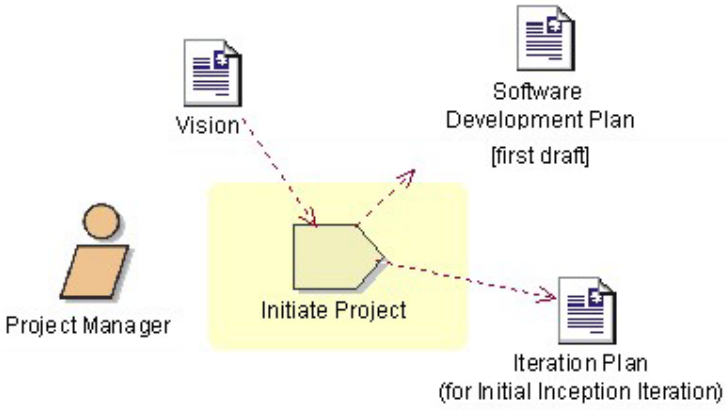
6.2.4 Project Management Workflow



Figur 6.2.4.1 *Project Management Workflow*

En release kan bestå av flera delprojekt som genomförs separat. Projektledaren rapporterar till produktledaren. Hur fyllig dokumentationen görs är upp till varje organisation och endast ett fåtal dokument och aktiviteter är borttagna. En del av dokumenten kan med fördel efter hand upprättas som policy-dokument för organisationen. Granskningsaktiviteterna ska uppfattas som beslutspunkter och behöver inte nödvändigtvis vara formella. Detta är en avvägningsfråga för organisationen.

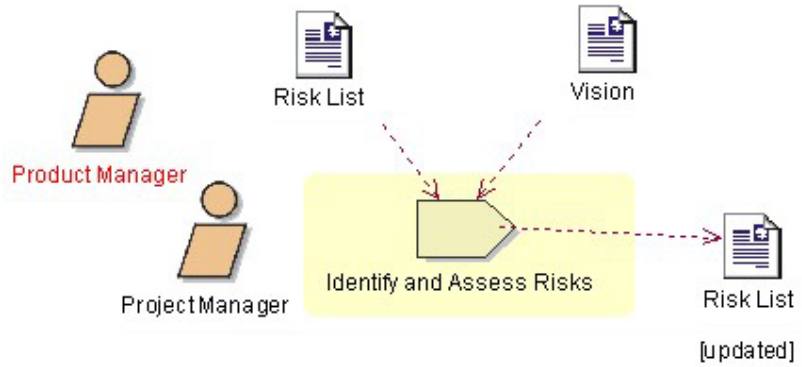
6.2.4.1 Conceive New Project



Figur 6.2.4.1.1 *Conceive New Project*

- Initiate Project:
  Källa: RUP.
  - Tillsätt projektplaneringsteam.
  - Första iterationsplan.
  - Arbeta vidare på SDP.

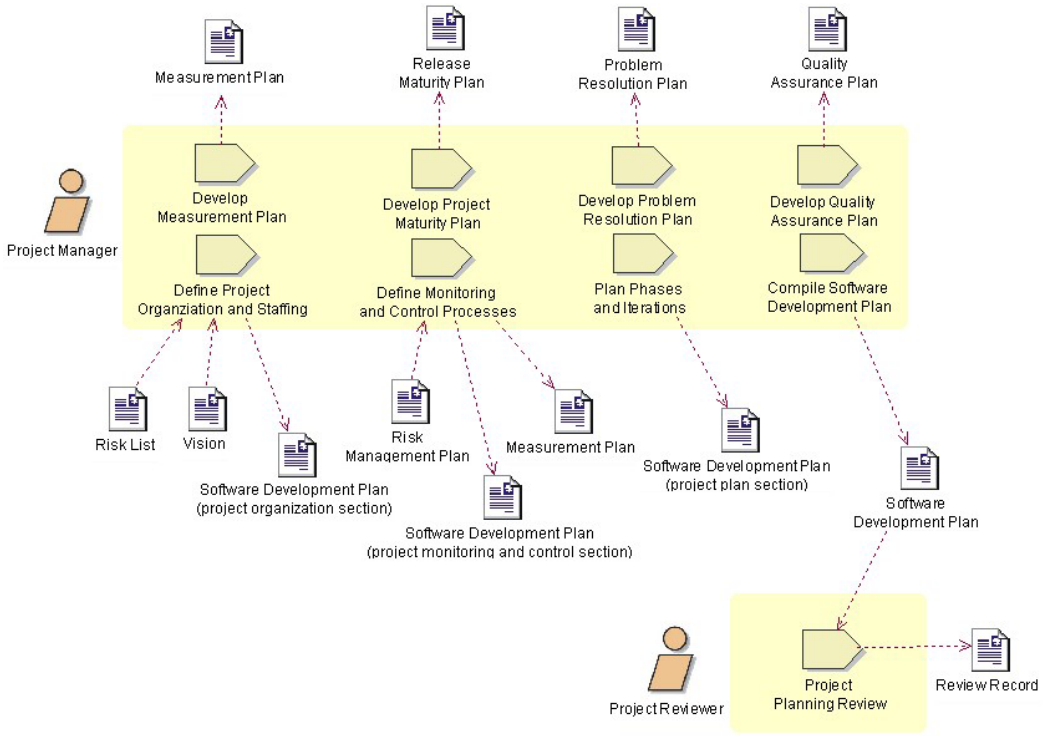
6.2.4.2 Evaluate Project Scope and Risk



Figur 6.2.4.2.1 *Evaluate Project Scope and Risk*

- Identify and Assess Risks:
  Källa: RUP
  - Identifiera potentiella risker (resursrisker, affärsrisker, tekniska risker, schemarisker)
  - Analysera och prioritera risker
  - Identifiera strategier för att undvika och begränsa risker

6.2.4.3 Develop Software Development Plan



Figur 6.2.4.3.1 *Develop Software Development Plan*

Hur fyllig dokumentationen görs är en avvägning för varje organisation. En del dokument kan ses som policy-dokument för organisationen. Andra kan vara delkapitel i en övergripande SDP.

- Develop Measurement Plan:
  Görs eventuellt inte inför varje projekt, utan kan ingå i företagets gemensamma rutiner. Källa: RUP.
  - Definiera och validera ledningsmål i termer av kvalitet, framåtskridande och förbättringar.
  - Besluta vad som ska mätas regelbundet för att stödja dessa mål (primära mätdata), och sätt insamlingsmekanismer på plats.

- Develop Project Maturity Plan:
  Specificera vad som ska vara klart till deadline, hur acceptansen av projektet ska gå till och vem som godkänner. Källa: RUP.

- Develop Problem Resolution Plan:
  Källa: RUP.
  - Definiera procedurer för att lösa specificerade problem.
  - Välj metoder och tekniker för att spåra problem.

- Develop Quality Assurance Plan:
  Källa: RUP.
  - Se till att kvalitetsmål är definierade för projektet.

  - Definiera kvalitetssäkrande roller och ansvar.
  - Koordinera de olika planerna.
  - Definiera kvalitetssäkrande uppgifter och schemalägg.

- Define Project Organization and Staffing:
  Källa: RUP.
  - Definiera en organisatorisk struktur för projektet
  - Baserat på tidsuppskattningarna, definiera krav på personalen inför nästa iteration (noggrant): antal, bakgrund och erfarenhetsnivå. Definiera också inför kommande iterationer, fast med lägre noggrannhetsnivå, för att kunna börja rekrytera.

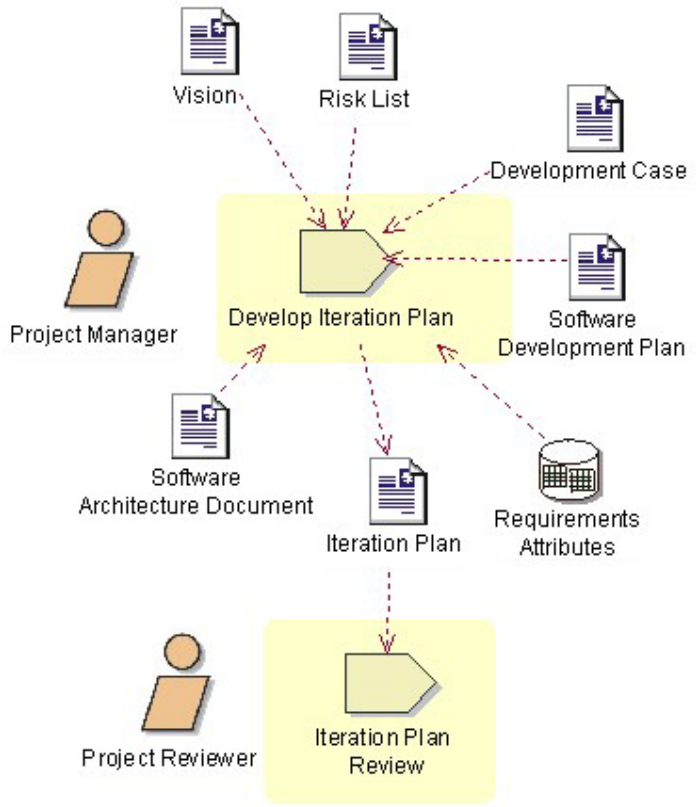
- Define Monitoring and Control Processes:
  Ev. görs inte detta till varje projekt. Kostnadsbevakning centralt. Källa: RUP.
  - Definiera information och processer som kommer att användas för att övervaka och kontrollera projektets framåtskridande, kvalitet och risker.
  - Definiera procedurer för att rapportera utvecklingsgruppens och projektets status.
  - Definiera procedurer och trösklar för att vidta åtgärder.

- Plan Phases and Iterations:
  Källa: RUP.
  - Uppskatta total omfattning, tidsåtgång och kostnad för projektet.
  - Utveckla en grovkornig projektplan som fokuserar på större milstolpar och projektets nyckelleverabler.
  - Definiera en uppsättning iterationer inom projektfaserna och identifiera mål och längd för varje iteration.
  - Utveckla schema och budget för projektet.
  - Utveckla en resursplan för projektet.
  - Definiera aktiviteter för att avsluta projektet på ett ordnat sätt.

- Compile Software Development Plan:
  Källa: RUP.
  - Koordinera utvecklingen av alla ingående planer och resterande innehåll för publicering i en övergripande Software Development Plan.

- Project Planning Review:
  Mer eller mindre formell granskning. Görs för att garantera att underlaget för att gå vidare med projektet är tillräckligt bra. Källa: RUP.
  - Granska och godkänn Software Development Plan.

6.2.4.4 Plan for Next Iteration

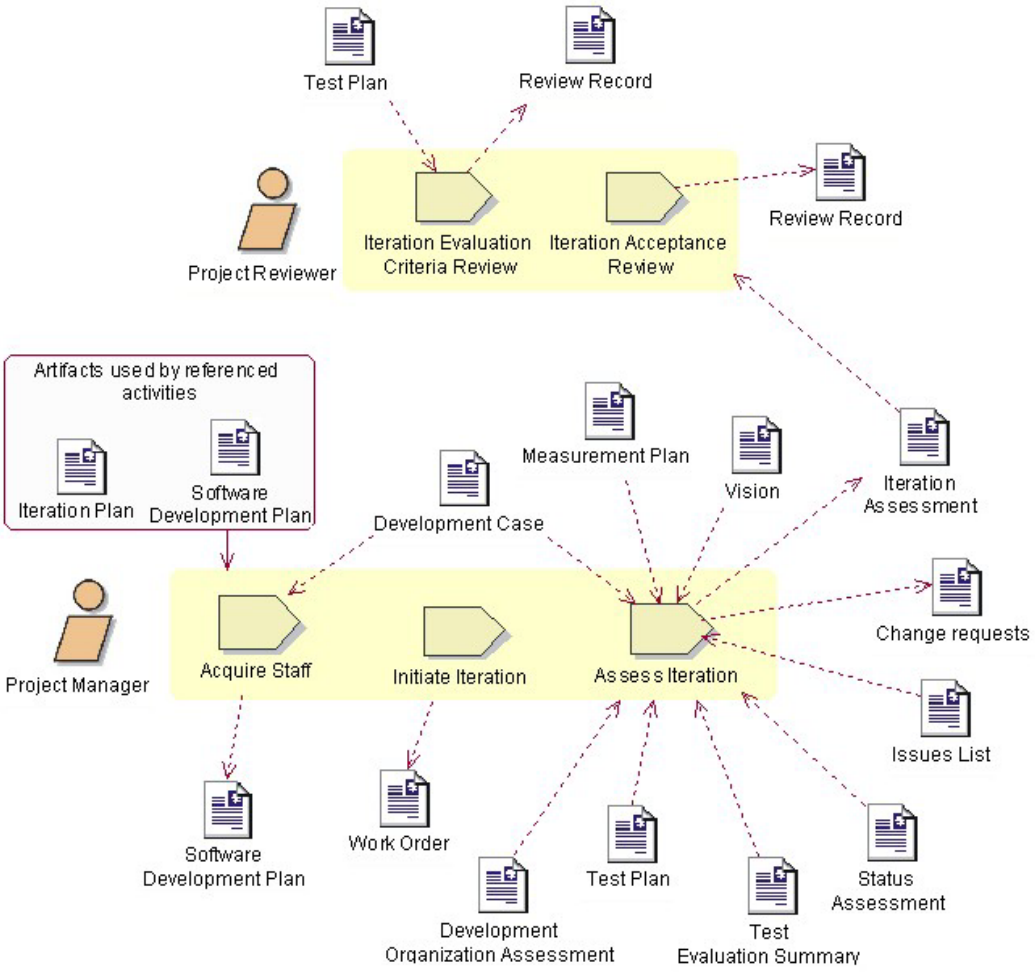


Figur 6.2.4.4.1 *Plan for Next Iteration*

- Develop Iteration Plan:
  Utveckla en detaljplan för en iteration. Källa: RUP.
  - Den ska innehålla en detaljplan över arbetet nedbrutet i aktivitets- och ansvarsuppgifter
  - Den ska innehålla milstolpar inuti iterationen och leverabler.

- Iteration Plan Review:
  Godkänn planeringen av arbetet för den kommande iterationen. Källa: RUP.

6.2.4.5 Manage Iteration



***Figur 6.2.4.5.1** Manage Iteration*

- Iteration Evaluation Criteria:
  Godkänner de kriterier som satts upp för att värdera om iterationen är att anse som klar. Källa: RUP.

- Iteration Acceptance Review:
  Ett formellt godkännande av att iterationen är att anse som klar. Källa: RUP.

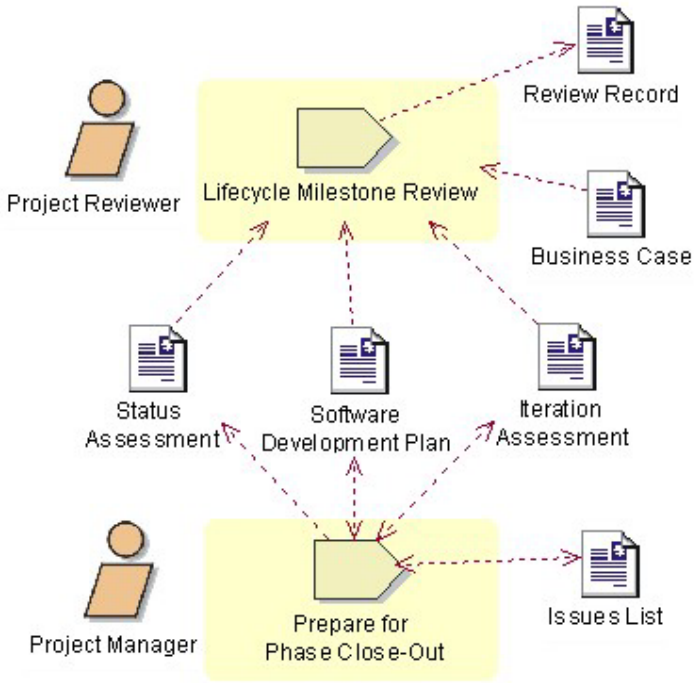
- Acquire Staff:
  Källa: RUP.
  - Tilldela projektet personal och koppla till rätt kunskapsområden.
  - Gruppera och utbilda resurserna så de blir relativt självständiga men samarbetande team.

- Initiate Iteration:
  Källa: RUP.
  - Tilldela personal till det arbete som ska utföras i iterationen (work packages).

- Skaffa och tilldela icke-personella resurser.
- Utfärda Work Orders.

- Assess Iteration:
  Källa: RUP.
  - Samla in mätdata.
  - Bestäm om iterationen är att anse som lyckad.
  - Utfärda Change Requests.

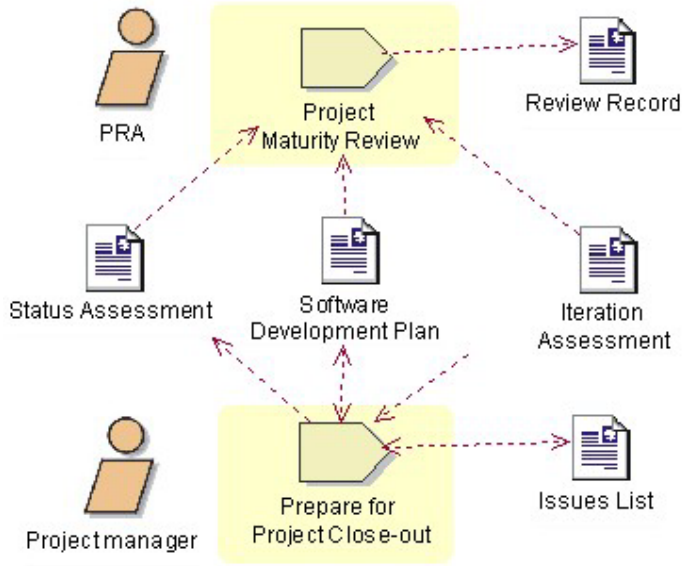
6.2.4.6 Close-Out Phase



Figur 6.2.4.6.1 *Close-Out Phase*

- Prepare for Phase Close-out:
  Källa: RUP.
  - Förbered material för Lifecycle Milestone Review.
  - Förbered projektet för att gå in i en ny fas.

6.2.4.7 Close-Out Project



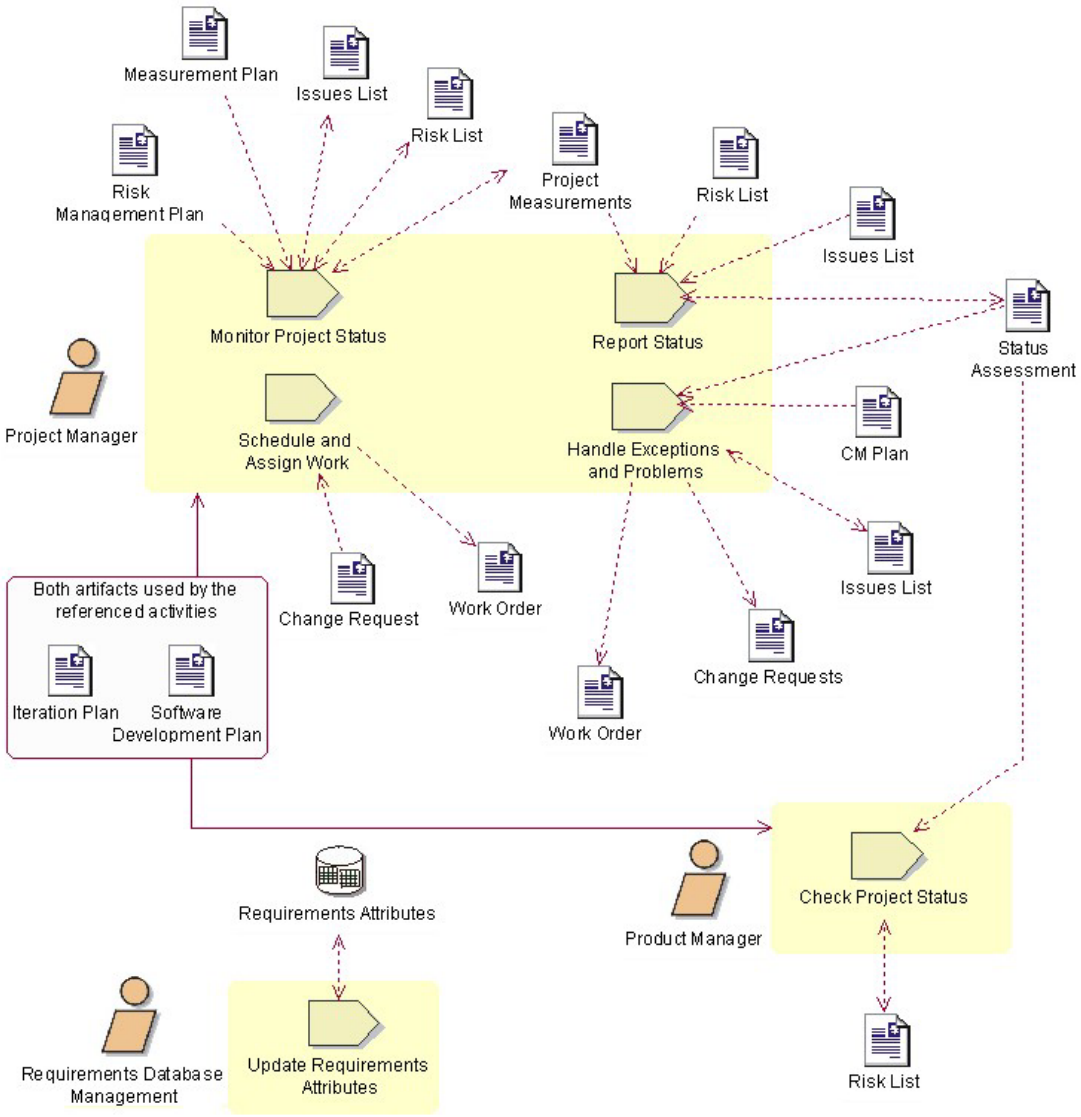
Figur 6.2.4.7.1 *Close-Out Project*

Eftersom MRUP hanterar delprojekt kommer releaser som inte består av ett enda projekt att behöva sättas samman. Vid slutet av varje delprojekt görs då en integrering med övrig kod, eller sammanslagning med annat projekt.

- Prepare for Project Close Out:
  Om projektet drivs som delprojekt, integrera med övrig release, eller med andra delprojekt, beroende på vad som planerats. Källa: utvärderingar.

- Project Maturity Review:
  PRA godkänner att projektet är att anses som klart.

6.2.4.8 Monitor and Control Project



***Figur 6.2.4.8.1** Monitor and Control Project*

- Monitor Project Status:
  Källa: RUP.
  - Ta in information om projektets status och utvärdera

- Report Status:
  Källa: RUP, men omarbetad då Product Manager tillkommit.
  - Rapportera regelbundet projektets status till Product Manager.
  - Låt Product Manager ta ställning till frågor som inte projektledaren har befogenhet att avgöra.
  - Gäller Project Manager.

- Schedule and Assign Work:
  Källa: RUP.
  - Allokera Change Request till en iteration.

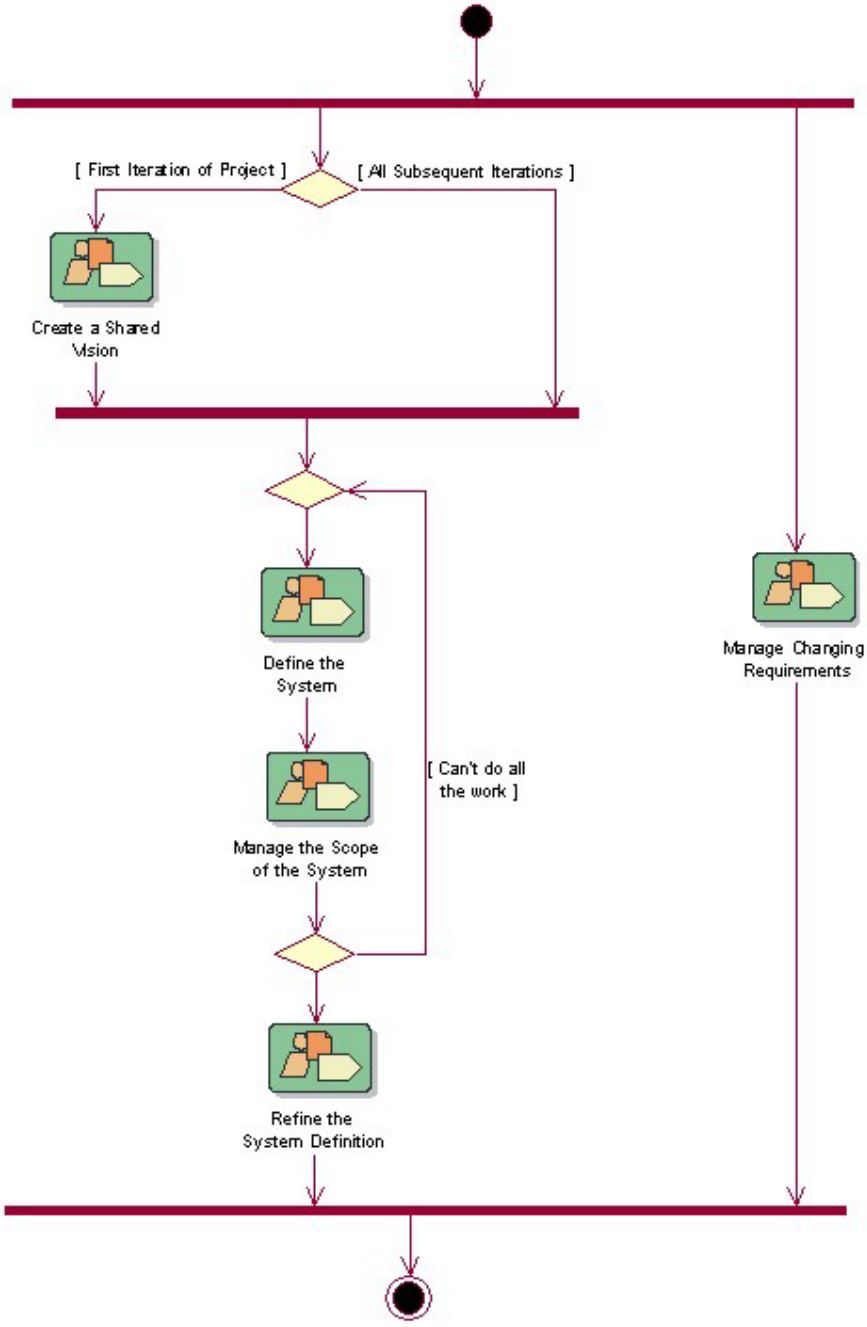
  - Tilldela ansvar.
  - Beskriv arbete och förväntat resultat.
  - Budgetera tidsåtgång och andra resurser.
  - Schemalägg och utfärda Work Order.

- Handle Exceptions and Problems:
  Källa: RUP.
  - Hantera de problem som uppstår.

- Check Project Status:
  Källa: fokusgruppmöten.
  - Samla ihop all information från de olika utvecklingsteamen och gör en samlad bedömning. Gäller Product Manager.

- Update Requirements Attributes:
  Detta arbete är viktigt för att alla icke releaseplanerade krav ska kunna utgöra ett vettigt material för att planera nästa release. Källa: utvärderingar.
  - Uppdatera statusattributet för implementerat krav.
  - För krav som blir inaktuella pga. att ett visst krav blivit implementerat, sätt statusattributet till *rejected*.

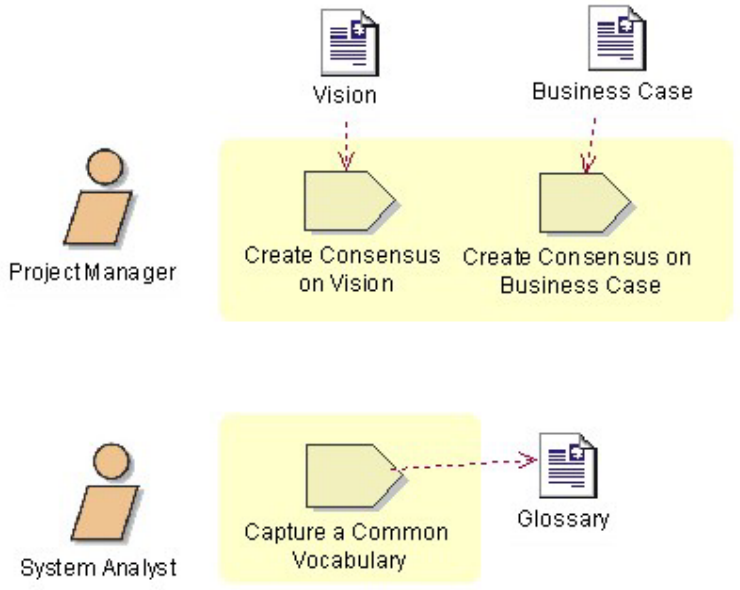
6.2.5 Requirements Workflow



Figur 6.2.5.1 *Requirements Workflow*

Arbetsflödet *Requirements* får in *Vision*-dokumentet med de högnivåkrav som ska implementeras. Till att börja med är det viktigt att skapa en gemensam vision om vilka produktens kunder och användare är för att utvecklarna lättare ska kunna fatta löpande beslut om hur systemet ska implementeras. I arbetsflödet hanteras utveckling av krav på detaljnivå. Förändringar för högnivåkrav hanteras som *Change Requests*. Observera att förändringar av uppsättningen av högnivåkrav får göras utan att genomföra aktiviteten *Release Planning* med efterföljande godkännande av releaseplanen.

6.2.5.1 Create a Shared Vision



Figur6.2.5.1.1 *Create a Shared Vision*

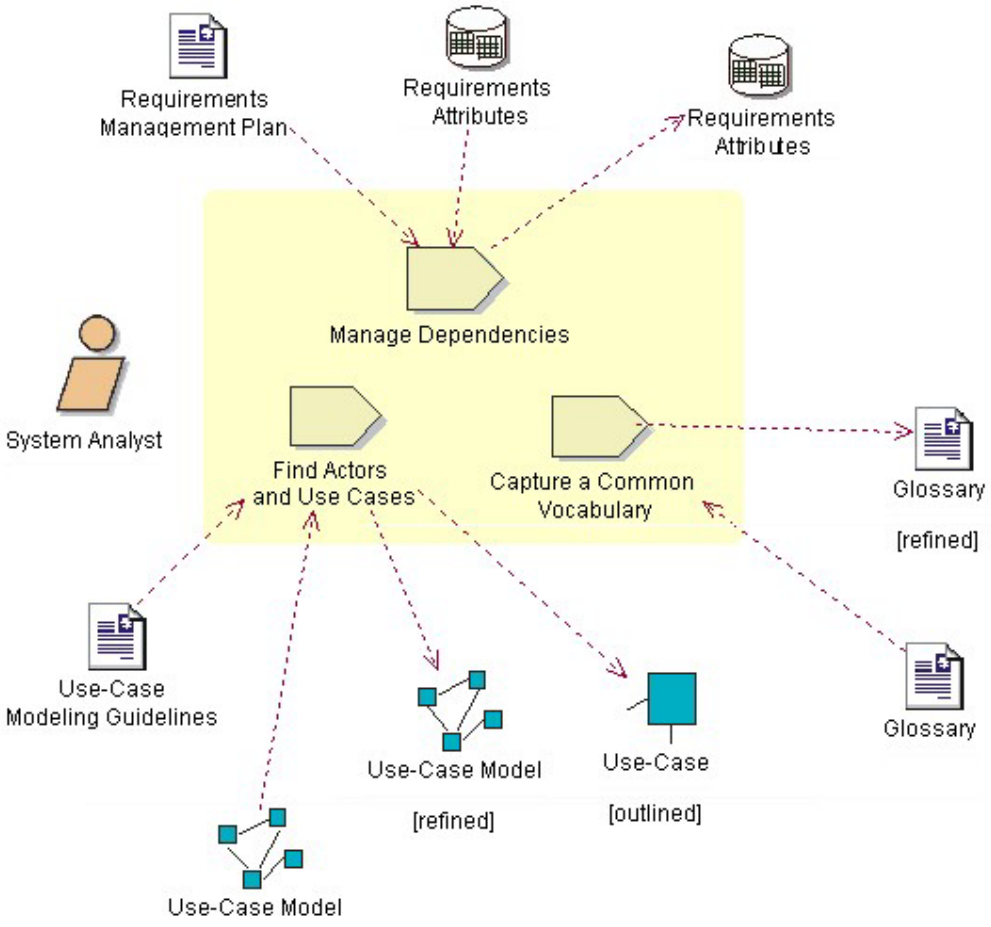
För att alla löpande beslut som fattas av enskilda utvecklare ska underlättas inleds arbetsflödet med att se till att alla har samma uppfattning om marknaden för produkten och för vem den utvecklas. Det ska också vara klart för alla varför produkten affärsmässigt är en bra idé.

- Create Consensus on Vision:
  Källa: Hooks & Farry, s 43.
  - Se till att alla i utvecklingsteamet har samma vision för produkten, mål för produktpositioneringen och produktens avsedda marknadssegment.
  - Se till att visionen är densamma som för marknads- och försäljningsavdelningen.

- Create Consensus on the Business Case:
  - Se till att alla i projektet förstår varför satsningen är lönsam.

- Capture a Common Vocabulary:
  Källa: RUP.
  - Hitta gemensamma termer och utvärdera resultatet.

6.2.5.2 Define the System



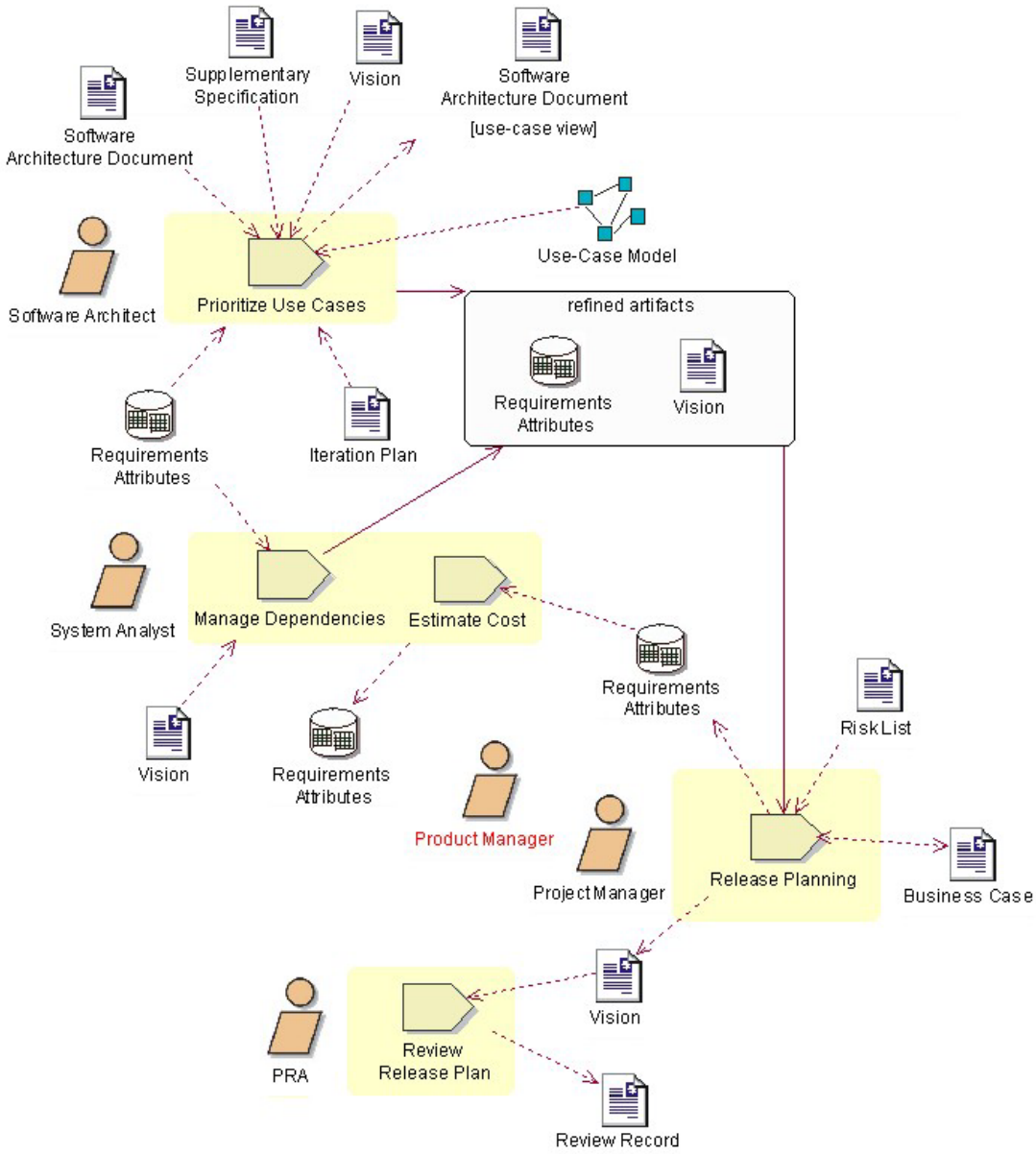
Figur 6.2.5.2.1 *Define the System*

- Manage Dependencies:
  Källa: RUP.
  - Tilldela attribut, etablera och verifiera spårbarhet.
  - Hantera ändringar av krav.

- Capture a Common Vocabulary:
  Källa: RUP.
  - Hitta gemensamma termer och utvärdera resultatet.

- Find Actors and Use Cases:
  Källa: RUP.
  - Hitta aktörer och användningsfall och beskriv hur de interagerar.
  - Skapa paket av användningsfall och aktörer.
  - Presentera användningsfallsmodellen i användningsfallsdiagram.
  - Granska och utvärdera.

6.2.5.3 Manage the Scope of the System



Figur 6.2.5.3.1 *Manage the Scope of the System*

- Prioritize Use Cases:
  Obs! Prioritering inför implementering. Källa: RUP.
  - Prioritera användningsfall och scenarion.

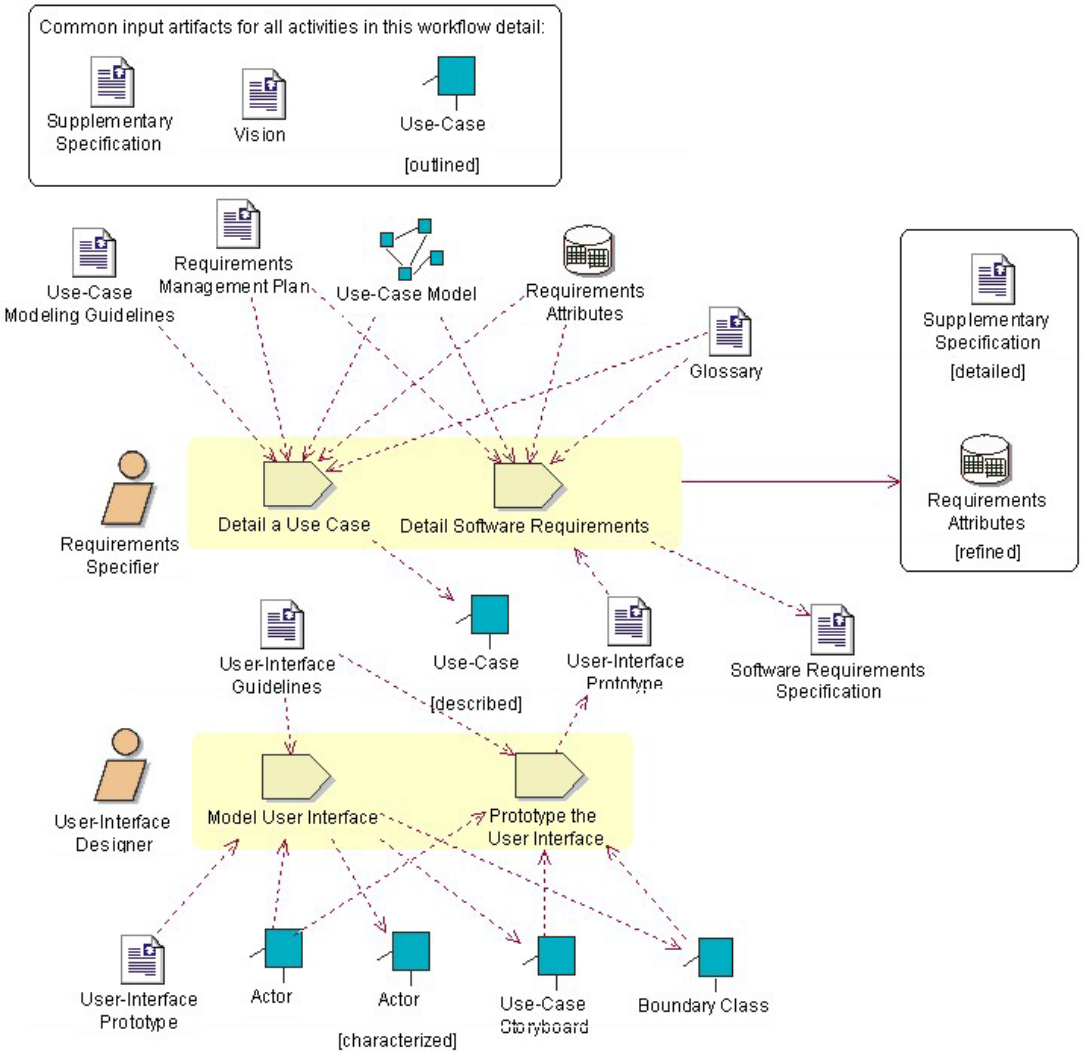
- Manage Dependencies:
  Källa: RUP.
  - Tilldela attribut, etablera och verifiera spårbarhet.
  - Hantera ändringar av krav.

- Estimate Cost:
  Detaljerad kostnadsuppskattning på valfritt sätt, t ex med hjälp av Lichtenbergmetoden eller expertbedömning. Se kap 6.2.3.3.

- Release Planning:
  Se kap 6.2.3.3. Bör göras om releasens innehåll blir påverkat av gränsdragningar eftersom nu underlaget om kostnader och risker är mer detaljerat.

- Review Release Plan:
  Se kap 6.2.3.3. Anledningen är att det är viktigt att inte göra ändringar utan att vara medveten om vilka effekter det får kundvärdet.

6.2.5.4 Refine the System Definition



Figur6.2.5.4.1 *Refine the System Definition*

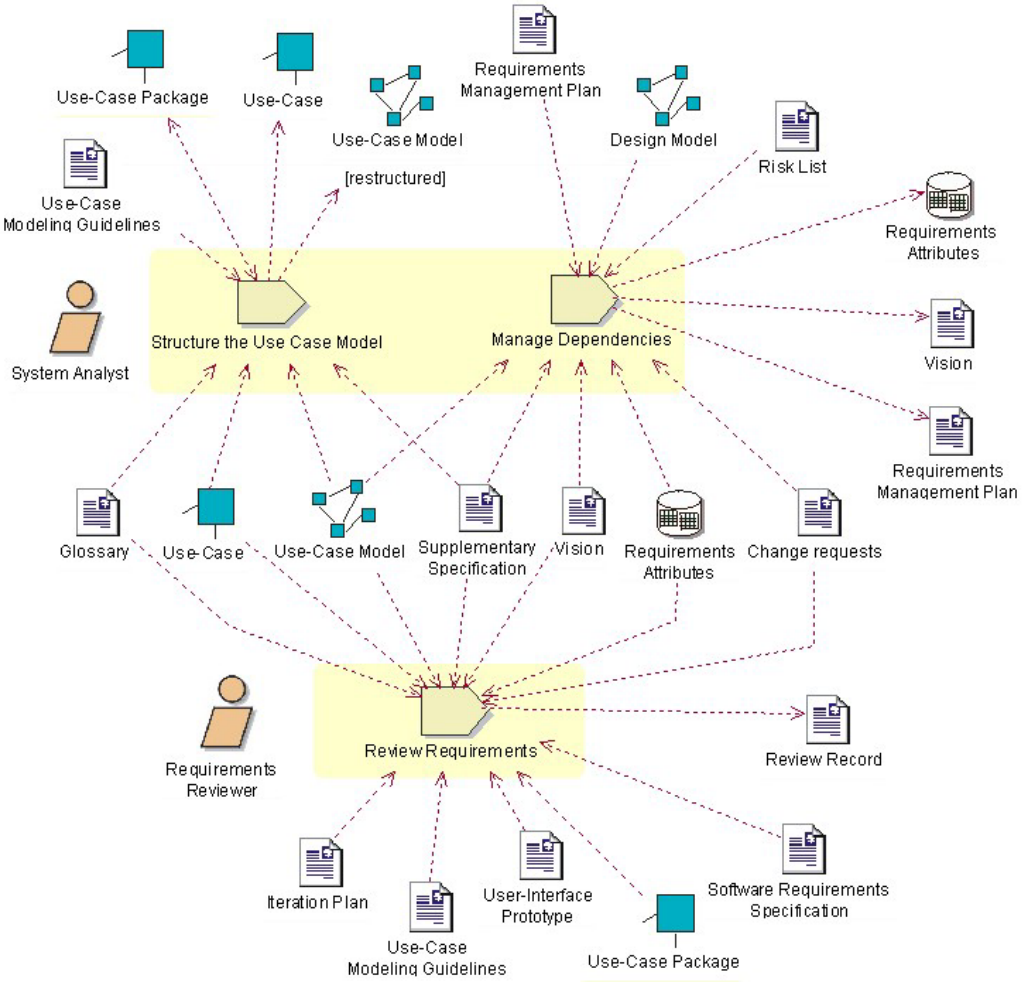
- Detail a Use Case:
  Källa: RUP.
  - Beskriv och strukturera i detalj flödet för användningsfallet.
  - Illustrera relationen med aktörer och andra användningsfall.
  - Beskriv specialkrav för användningsfallet
  - Beskriv kommunikationsprotokoll.
  - Beskriv villkor för att starta och avsluta användningsfallet (ej nödvändigt).
  - Beskriv utökningspunkter (ej nödvändigt)
  - Utvärdera resultatet.

- Detail the Software Requirements:
  Källa: RUP.
  - Detaljera och paketera programvarukraven.

- Model the User Interface:
  Källa: RUP.
  - För alla användningsfall som blivit prioriterade ur ett usabilityperspektiv, modellera ett användargränssnitt med hjälp av storyboards.

- Prototype the User Interface:
  Källa: RUP.
  - Designa, implementera och få feedback på användargränssnittet.

6.2.5.5 Manage Changing Requirements



Figur 6.2.5.5.1 *Manage Changing Requirements*

- Structure the Use Case Model:
  Källa: RUP.
  - Etablera inkluderingar och utvidgningar mellan användningsfall, och generaliseringar mellan användningsfall och mellan aktörer.

- Manage Dependencies:
  Källa: RUP.
  - Tilldela attribut, etablera och verifiera spårbarhet.
  - Hantera ändringar av krav.

- Review Requirements:
  Källa: RUP. Granskningar kan handla om tre olika sorter:
  - Granskning av hur Change Requests påverkar nuvarande kravuppsättning
  - Granskning av hela användningsfallsmodellen
  - Granskning av användningsfallen inklusive diagram.

Kapitel 7

Slutsats

MRUP är ett förslag till ett marknadsdrivet alternativ till RUP. Det behöver fortfarande valideras i riktiga projekt och byggas vidare. Vi har mött ett stort intresse för vårt arbete på vägen, både bland våra informanter och i övriga programvaruindustrin. Slutsatsen blir att vårt förslag verkar lovande. Eftersom i stort sett varje kommersiellt programvaruutvecklande företag själv uppfinner sina egna metoder, finns det absolut ett hål att fylla för företag som inriktat sig på utveckling av processtöd. Det finns ju ingen anledning att uppfinna hjulet alltför många gånger, även om justeringar kan krävas för att få det att passa.

Den allmänna kritiken som riktats mot RUP är att det är för stort och det är svårt att veta i vilken ände man ska börja införa processtöden. Vår rekommendation, förutsatt att man redan har någon typ av process för sin utveckling, är att först satsa på hanteringen av kravdatabasen. Tillsätt någon eller några som löpande tar hand om inkommande krav, och som gör en effektiv screening av dem och ser till att de som läggs in i databasen blir tidsuppskattade. Se också till att personen eller gruppen har resurser och tid. På så sätt görs förarbetet inför nästa stora releaseplanering och det finns bättre möjlighet att se när ett riktigt *showstopper*-krav kommer in och som bör planeras till pågående release. Det andra viktiga området är prioriteringen, särskilt med en metod som värdeanalys med visualisering av resultatet. Det blir då lättare att se hur nöjt varje kundsegment blir med en viss uppsättning features. En tredje viktig aktivitet är att se över hela sin releaseplan varje gång förändringar i den görs. Den typen av beslut har affärsmässiga konsekvenser och besluten måste tas med öppna ögon. Även där gör visualiserad värdeanalys det lättare att se hur förändringar påverkar helheten. Övriga aktiviteter i *Release Planning Workflow* kan plockas in efter hand, men aktiviteter som rör riskhantering kommer ofta långt ner på prioriteringsskalan för företagen.

Kapitel 8

Vidare forskning och arbete

Förslaget behöver genomgå validering i riktiga projekt. På så sätt skulle mer detaljer utkristallisera sig i fråga om kravhanteringen, men också om det marknadsarbete som görs. En möjlighet är att testa enbart *Release Planning Workflow* i ett mindre projekt, där någon form av RUP-anpassning används i övrigt. Ändringarna i de två övriga arbetsflödena är ju till stor del konsekvenser av *Release Planning* och det är troligen lättare att få tillgång till projekt i den änden av arbetet. En möjlighet är också att implementera detta arbetsflöde som en plug-in till övriga RUP och manuellt anpassa *Requirements Workflow* och *Project Management Workflow*.

Det finns några områden som också borde ses över, men som av tidsbrist inte tittats närmare på. MRUP är nu ganska fokuserad på funktionella krav vid värdeanalys och releaseplanering. Eventuellt borde icke-funktionella krav och arkitekturfrågor uttalat behandlas i samband med releaseplaneringen. Ett annat område är prototyphantering som för kommersiella företag kan vara ett bra sätt att få marknadens feedback med. Särskilt gäller det utvecklingen av nya produkter, då utvecklingen av krav är svår att göra på ett så pass dokumentstyrt processramverk som RUP är.

Som nämnts tidigare pågår mycket forskning inom marknadsdriven kravhantering och många besvärliga områden skulle behöva metodik för att klaras av. Beroenden är till exempel ett område som komplicerar releaseplaneringen mycket, och mer forskning skulle behövas för att hantera det mer strukturerat.

Referenser

[1] Berelsen, B. (1952). *Content Analysis in Communications Research*. New York, Free Press.

[2] Boehm, B. W. (1988). A Spiral Model of Software Development and Enhancement, *Computer*, vol 21, nr 5, s 61-72, Los Alamitos, IEE/IEEE.

[3] Boehm, B. W. (1979). Software engineering; R & D trends and defense needs. In *Research Directions in Software Technology* (P. Wegner, ed.). Cambridge , MIT Press (kap 19).

[4] Brooks, F. P. (1995). *The Mythical Man-Month*. Anniversary Edition. Reading, Addison-Wesley.

[5] Carlshamre, P. (2002). Release Planning in Market-driven Software Product Development - Provoking an Understanding. *Requirements Engineering Journal* 7(3):139-151.

[6] Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., Natt och Dag, J. (2001) An Industrial Survey of Requirements Interdependencies in Software Release Planning. *Proc. Fifth IEEE Int. Symposium on Requirements Engineering* (RE'01), Aug. 27-31, Toronto, Canada.

[7] Carlshamre, P., Regnell, B. (2000). Requirements Lifecycle management and Release Planning in Market-Driven Requirements Engineering Processes. Publicerad vid *International Workshop on the Requirements Engineering Process: Innovative Techniques, Models and Tools to support the RE Process, 6yh-8th of September 2000, Greenwich UK*. IEEE CS Press.

[8] Carlshamre, P. (2001) *A Usability Perspective on Requirements Engineering – Frp, Methodology to Product Development* (Dissertation No. 726). Linköping, Linköping University, Linköping Studies in Science and Technology.

[9] Clements, P., Northrop, L. (2002). *Software Product Lines. Practices and Patterns*. Boston, Addison Wesley.

[10] Davis, A. M. (1993). *Software Requirements – Objects, Functions, &States* (Rev. ed.). Upper Saddle River, Prentice Hall.

[11] Faulk. S., Harmon, R., Raffo, D. (2000). Value-Based Software Engineering (VBSE): A Value-Driven Approach to Product-Line Engineering, I *Software Product Lines: Experience and Research Direction* (Donohoe ed). Denver, Colorado. Boston, Kluwer Academic Publishers.

[12] Grady, R. (1992). *Practical Software Metrics for Project Management and Process Improvement*. Englewood Cliffs, NJ, Prentice-Hall.

[13] Hirsch, M. (2002). Making RUP Agile. *Conference on Object Oriented Programming Systems Languages and Applications OOPSLA 2002 Practitioners Reports*. ACM Press New York, NY, USA

[14] Höst, M., Regnell, B., Natt och Dag, J., Nedstam, J., Nyberg, C. (2001). Exploring bottlenecks in Market-Driven Requirements Managements Processes with Discrete Event Simulations. In *Journal of Systems and Software, 59*, s 323-332.

[15] Höst, M., Wohlin, C. (1998). An Experimental Study of Individual Subjective Effort Estimations and Combinations of the Estimates. In *Proceedings the 20th International Conference on Software Engineering, ICSE 98, April 19-25, 1998, Kyoto, Japan* (s 332-339), Los Alamitos, IEEE Computer Society.

[16] IEEE Standard 1233-1998 (2000). IEEE guide for developing system requirements specifications. In *Software Requirements Engineering, Second Edition,* R. H. Thayer and M. Dorfman, (Eds.) (s 245-280). Los Alamitos, IEEE Computer Society.

[17] IEEE 610.12-1990. Standard Glossary of Software Engineering Terminology.

[18] Karlsson, J., (1998). *A Systematic Approach for Prioritizing Software Requirements* (Dissertation no. 526). Linköping: Linköping University, Linköping Studies in Science and Technology.

[19] Karlsson, J., Wohlin, C., Regnell, B. (1997). An Evaluation of Methods for Prioritizing Software Requirements. *Journal of Information and Software Technology,* Vol. 39, Nr. 14-15, s. 939-947.

[20] Karlsson, J. (2002). *Marknadsdriven produktledning – från kundbehov och krav till lönsamma produkter* Linköping, Focal Point AB.

[21] Kotler, P., Armstrong, G., Saunders, J., Wong, V. (2002). *Principles of Marketing.* Upper Saddle River, N.J., Prentice Hall.

[22] Kotonya, G., Sommerville, I. (1998). *Requirements Engineering: Processes and Techniques*, Chichester, John Wiley & Sons Ltd.

[23] Lausen, S. (2002). *Software Requirements. Styles and Techniques.* Pearson Education Limited, Addison Wesley.

[24] Lehmann, D.R., Winer, R.S. (2002). *Product Management.* Boston, McGraw-Hill.

[25] Lewis, C., Rieman, J., (1994) *Task-centered User Interface Design. A Practical Introduction.* http://www.technosphere.net/tcuid/tcuid.htm.
Senast kontrollerad 2003-11-16.

[26] Lichtenberg, S. (2000). *Proactive management of Uncertainty Using the Successive Principle. A practical way to manage opportunities and risks.* Copenhagen, Polyteknisk Press.

[27] Lubars, M., Potts, C., Richter, C. (1993). A Review of the State of the Practice in Requirements Modeling. *Proceedings of First IEEE International Symposium on Requirements Engineering.* Los Alamitos, CA: IEEE Computer Society Press.

[28] Maiden, N.A.M., Rugg, G. (1996). ACRE:Selecting Methods for Requirements Acquisition. *Software Engineering Journal,* 11(3):183-192. London, IEEE.

[29] Moore, G. A. (1999). *Crossing the Chasm: Marketing and Selling High-Tech Products to Mainstream Customer.* New York, Harper Business.

[30] Moore, G. A. (1998). *Inside the Tornado. Marketing Strategies from Silicon Valley's Cutting Edge.* Oxford, Capstone.

[31] Natt och Dag, J. (2002). *Elicitation and Management of User Requirements in Market-Driven Software Development* Lund, Lund University, Department of Communication Systems, Lund Institute of Technology.

[32] Novorita, R. J., Grube, G. (1996). Benefits of Structured Requirements Methods for Market-Based Enterprises. *Proceedings of International council on Systems Engineering Sixth Annual International Symposium on Systems Engineering: Practice and Tools (INCOSE'96),* Boston, USA.

[33] Potts, C. (1995). Invented Requirements and Imagined Customers: Requirements Engineering for Off-theShelf Software. In *Proceedings of the Second IEEE International Symposium on Requirements Engineering* (pp.128-130). Los Alamitos, CA: IEEE Computer Society Press.

[34] Rational Software (2002). *Rational Unified Process version 2002.05.00.*

[35] Regnell, B. (1998). *Requirements Engineering with Use Cases – a Basis for Software Development.* Lund, Lund Univerisity, Department of Communication Systems, Lund Institute of Technology.

[36] Regnell, B., Beremark, P., Eklundh, O. (1998). Market-driven Requirments Engineering Process. Results from an Industrial Process Improvement Programme. In *Requirements Engineering* 3 (s 121-129). Springer-Verlag.

[37] Regnell, B., Karlsson, L., Höst, M. (2003) An Analytical Model for Requirements Selection Quality Evaluation in Product Software Development. *RE '03 – IEEE 11th*

International Conference on Requirements Engineering, September 8-12, Monterey Bay, California, USA, 2003.

[38] Robson, C. (2002). *Real World Research* (2nd ed.). Oxford, Blackwell.

[39] Sawyer, P., Sommervill, I., Kotonya, G. (1999). Improving Market-Driven RE Processes. In M. Oivo, & P.Kuvaja (Eds.) *Proceedings from International Conference on Product Focused Software Process Improvement* (s 222-236). Oulo, Finland, Technical Research Centre of Finland (VTT).

[40] Trochim, W. (2002). *The Research Methods Knowledge Base.* http://trochim.human.cornell.edu/kb/index.htm Senast ändrad 2003-06-08. Senast kontrollerad 2003-11-16.

[41] Yeh A.C. (1992). Requirements engineering support technique (REQUEST): a market driven requirements management process**.** *Assessment of Quality Software Development Tools, 1992. Proceedings of the Second Symposium.* Los Alamitos, IEEE Computer Society Press.

[42] Young, R. R., (2001). *Effective Requirements Practices.* Boston, Addison-Wesley.

Ordlista

Business Actor	Verksamhetsaktör
Concept	Begrepp
Customer	Kund
Elicitation	Kravinsamling
Feature	Funktionellt högnivåkrav
Guidelines	Riktlinjer
Market nicher	Nischföretag
Market share	Marknadsandelar
Market targeting	Målmarknadsföring
Performance	Prestanda
Project Manager	Projektledare
Project Reviewer	Projektgranskare
Reliability	Tillförlitlighet
Requirement	Krav
Requirement conception	Kravens ursprung
Requirements Reviewer	Kravgranskare
Requirements Specifier	användningsfallsspecificerare
Software Architect	Programvaruarkitekt
Software Engineering	Programvaruutveckling
Specification	Specifikation
Stakeholder	Intressent
Supportability	Modifierbarhet
Supporting discipline	Stödjande disciplin
System Analyst	Systemanalytiker
Target market	Utvalt marknadssegment
Target marketing	Målmarknadsföring
Template	Malldokument
Tool Mentor	Verktygsguide
Usability	Användbarhet
Use case	Användningsfall
User-Interface Designer	Användargränssnittsdesigner
Use-Case Specifier	Användningsfallsspecificerare
Validation	Validering
Workflow	Arbetsflöde
Workflow detail	Arbetsflödesdetalj

Appendix A

<Product Name>
Vision

Version <1.0>

[Note: The following template is provided for use with the Rational Unified Process. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

Revision History

Date	Version	Description	Author
<dd/mmm/yy>	<x.x>	<details>	<name>

Table of Contents

Vision

1 Introduction

[The purpose of this document is to collect, analyze, and define high-level needs and features of the <<System Name>>*. It focuses on the capabilities needed by the stakeholders, and the target users, and* ***which*** *these needs are. The details of how the* <<System Name>> *fulfils these needs are detailed in the use-case and supplementary specifications.]*

[The introduction of the ***Vision*** *document provides an overview of the entire document. It should include the purpose, scope, definitions, acronyms, abbreviations, references, and overview of this* ***Vision*** *document.]*

1.1 Purpose

[Specify the purpose of this ***Vision*** *document****.****]*

1.2 Scope

[A brief description of the scope of this ***Vision*** *document; what Product(s) it is associated with and anything else that is affected or influenced by this document.]*

1.3 Definitions, Acronyms, and Abbreviations

[This subsection provides the definitions of all terms, acronyms, and abbreviations required to properly interpret the ***Vision*** *document. This information may be provided by reference to the project's Glossary.]*

1.4 References

[This subsection provides a complete list of all documents referenced elsewhere in the ***Vision*** *document. Identify each document by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]*

1.5 Overview

[This subsection describes what the rest of the ***Vision*** *document contains and explains how the document is organized.]*

2 Market Strategy

[The market strategy includes the strategies that guide marketing actions and development focus. Areas to consider is how to segment the market and to target a market with considerations to the competition create a positioning goal to strive for and an analysis of the key needs in different markets.]

2.1 Business Opportunity

[Briefly describe the business opportunity being met by this project.]

2.2 Market Segments

[This is actually the result of the market segmentation process and a presentation of the market segments worth pursuing. Only the result of those processes is relevant here.]

2.2.1 <Market Segment>

[Describe the market segment. Estimate market size and current market share, if any. Relate the information to the competition in this segment. Express if the product is targeted at a specific competitor. Estimate market potential for the product. Also state relevant business intelligence information on the competition. Explain customer strategy: is the focus on acquiring new customers, keeping current customers satisfied, increasing current customers usage, or dropping non profitable customers.]

2.2.1.1 Current Product Positioning

[What is the products current position. Relate to the competition. Product lifecycle stage, i.e. introduction, growth, maturity or decline.]

2.2.1.2 Product Position Goal

[Provide an overall statement summarizing, at the highest level, the unique position the product intends to fill in the marketplace. The following format may be used:]

For	*[target customer]*
Who	*[statement of the need or opportunity]*
The (product name)	*is a [product category]*
That	*[statement of key benefit; that is, the compelling reason to buy]*
Unlike	*[primary competitive alternative]*
Our product	*[statement of primary differentiation]*

2.2.1.3 Key Stakeholders and Users

[Describe each key stakeholder or user type. What we want is a short description of their interest as a stakeholder and their needs. State their over all level of experience in this kind of software solution (for example computer guru) Also describe their environment for example in terms of platforms in use, other applications in use which might require integration, mobility etc.]

2.2.2 <Another Market Segment>

3 Product Overview

[This section provides a high level view of the product capabilities, interfaces to other applications, and systems configurations. This section usually consists of three subsections, as follows:

- *Product perspective*
- *Product functions*
- *Assumptions and dependencies]*

3.1 Product Perspective

*[This subsection of the **Vision** document puts the product in perspective to other related products and the user's environment. If the product is independent and totally self-contained, state it here. If the product is a component of a larger system, then this subsection relates how these systems interact and identifies the relevant interfaces between the systems. One easy way to display the major components of the larger system, interconnections, and external interfaces is with a block diagram.]*

3.2 Summary of Capabilities

*[Summarize the major benefits and features the product will provide. For example, a **Vision** document for a customer support system may use this part to address problem documentation, routing, and status reporting without mentioning the amount of detail each of these functions requires.*

Organize the functions so the list is understandable to anyone reading the document for the first time. A simple table listing the key benefits and their supporting features might suffice. For example:]

Table 4-1 Customer Support System

Customer Benefit	Supporting Features
New support staff can quickly get up to speed.	Knowledge base assists support personnel in quickly identifying known fixes and workarounds.
Customer satisfaction is improved because nothing falls through the cracks.	Problems are uniquely itemized, classified and tracked throughout the resolution process. Automatic notification occurs for any aging issues.
Management can identify problem areas and gauge staff workload.	Trend and distribution reports allow high level review of problem status.
Distributed support teams can work together to solve problems.	Replication server allows current database information to be shared across the enterprise.
Customers can help themselves, lowering support costs and improving response time.	Knowledge base can be made available over the Internet. Includes hypertext search capabilities and graphical query engine.

3.3 Assumptions and Dependencies

*[List each of the factors that affect the features stated in the **Vision** document. List assumptions that, if changed, will alter the **Vision** document. For example, an assumption may state that a specific operating system will be available for the hardware designated for the software product. If the operating system is not available, the **Vision** document will need to change.]*

3.4 Cost and Pricing

[For products sold to external customers and for many in- house applications, cost and pricing issues can directly impact the application's definition and implementation. In this section, record any cost and pricing constraints that are relevant. For example, distribution costs (# of diskettes, # of CD-ROMs, CD mastering) or other cost of goods sold constraints (manuals, packaging) may be material to the projects success, or irrelevant, depending on the nature of the application.]

3.5 Licensing, Installation and Distribution

[Licensing and installation issues can also directly impact the development effort. For example, the need to support serializing, password security or network licensing will create additional requirements of the system that must be considered in the development effort.

Installation requirements may also affect coding or create the need for separate installation software.

Distribution requirements.]

4 Release Plan

4.1 <Next Release>

4.1.1 Key dates

[Give a list with all included features listed in order of priority.]

4.1.2 Features

[List and briefly describe the product features. Features are the high-level capabilities of the system that are necessary to deliver benefits to the users. Each feature is an externally desired service that typically requires a series of inputs to achieve the desired result. For example, a feature of a problem tracking system might be the ability to provide trending reports. As the use-case model takes shape, update the description to refer to the use cases.

*Because the **Vision** document is reviewed by a wide variety of involved personnel, the level of detail needs to be general enough for everyone to understand. However, enough detail must be available to provide the team with the information they need to create a use-case model.*

To effectively manage application complexity, we recommend for any new system, or an increment to an existing system, capabilities are abstracted to a high enough level so 25-99 features result. These features provide the fundamental basis for product definition, scope management, and project management. Each feature will be expanded in greater detail in the use-case model.

Throughout this section, each feature will be externally perceivable by users, operators or other external systems. These features should include a description of functionality, any relevant usability issues that must be addressed, any relevant dependencies with other feature and cost of implementation. The following guidelines apply:

- *Avoid design. Keep feature descriptions at a general level. Focus on capabilities needed and why (not how) they should be implemented.*
- *If you are using the Rational RequisitePro toolkit, all need to be selected as requirements of type for easy reference and tracking.]*

4.1.3 Value Analysis

4.1.3.1 Value Analysis<Internal value>

[The internal value could include marketing potential offered by the feature, or administrative features having potential business value, or technical advantages in implementing the feature. A group representing all these aspects performs the internal value analysis. Value analysis is a prioritization of the features performed through pair-wise comparison. Visualize.]

4.1.3.2 Value Analysis <A Market Segment>

[Value Analysis from the given market segments point of view. How do this market segment prioritize the features and how happy will this market segment be with this release plan?]

4.1.3.3 Value Analysis <Another Market Segment>

4.1.4 Other Release Requirements

[At a high-level, list applicable standards, hardware or platform requirements, performance requirements, and environmental requirements.]

4.1.4.1 Usability Requirements

[List high-level usability requirements that this release has to support.]

4.1.4.2 Applicable Standards

[List all standards with which the product must comply. These can include legal and regulatory (FDA, UCC) communications standards (TCP/IP, ISDN), platform compliance standards (Windows, UNIX, and so on), and quality and safety standards (UL, ISO, CMM).]

4.1.4.3 System Requirements

[Define any system requirements necessary to support the application. These can include the supported host operating systems and network platforms, configurations, memory, peripherals, and companion software.]

4.1.4.4 Performance Requirements

[Use this section to detail performance requirements. Performance issues can include such items as user load factors, bandwidth or communication capacity, throughput, accuracy, and reliability or response times under a variety of loading conditions.]

4.1.4.5 Environmental Requirements

[Detail environmental requirements as needed. For hardware- based systems, environmental issues can include temperature, shock, humidity, radiation, and so forth. For software applications, environmental factors can include usage conditions, user environment, resource availability, maintenance issues, and error handling and recovery.]

4.1.4.6 Quality Ranges

[Define the quality ranges for performance, robustness, fault tolerance, usability, and similar characteristics that are not captured in the Feature Set.]

4.1.4.7 Constraints

[Note any design constraints, external constraints or other dependencies.]

4.2 <The Release After That>

4.2.1 Features

[Value Analysis will not be necessary, just state features.]

4.2.2 Other Release Requirements

[If there are other requirements, these can be stated here.]

5 Projects

[The release might be developed within separate projects. Each requirement should have an attribute identifying which project is responsible. This section is a listing of the included projects. For each projects state the overall purpose.]

6 Documentation Requirements

[This section describes the documentation that must be developed to support successful application deployment. Anything that needs implementation should be noted as a feature in section 4.1.1.]

6.1 User Manual

[Describe the purpose and contents of the User Manual. Discuss desired length, level of detail, need for index, glossary of terms, tutorial versus reference manual strategy, and so on. Formatting and printing constraints must be identified also.]

6.2 Online Help

[Many applications provide an online help system to assist the user. The nature of these systems is unique to application development as they combine aspects of programming (hyperlinks, and so on) with aspects of technical writing such as organization and presentation. Many have found the development of online help system is a project within a project that benefits from up-front scope management and planning activity.]

6.3 Installation Guides, Configuration, and Read Me File

[A document that includes installation instructions and configuration guidelines is important to a full solution offering. Also, a Read Me file is typically included as a standard component. The Read Me file can include a "What's New With This Release" section, and a discussion of compatibility issues with earlier releases. Most users also appreciate documentation defining any known bugs and workarounds in the Read Me file.]

6.4 Labeling and Packaging

[Today's state-of-the-art applications provide a consistent look and feel that begins with product packaging and manifests through installation menus, splash screens, help systems, GUI dialogs, and so on. This section defines the needs and types of labeling to be incorporated into the code. Examples include copyright and patent notices, corporate logos, standardized icons and other graphic elements, and so forth.]

7 Support

[What kind of support is connected to the product? If new specific software is needed, note as feature in section 4.1.1.]

8 Services

[What services are connected to the product? If new specific software is needed, note as feature in section 4.1.1.]

A Feature Attributes

[Features are given attributes that can be used to evaluate, track, prioritize, and manage the product items proposed for implementation. All requirement types and attributes are outlined in the Requirements Management Plan, however you may wish to list and briefly describe the attributes for features that have been chosen. The following subsections represent a set of suggested feature attributes.]

A.1 Reason

[This text field is used to track the source of the requested feature. Requirements exist for specific reasons. This field records an explanation or a reference to an explanation. For example, the reference might be to a page and line number of a product requirement specification or to a minute marker on a video of an important customer review.]

A.2 Status

[Set after negotiation and review by the project management team. Tracks progress during definition of the project baseline.]

Proposed	*[Used to describe features that are under discussion but have not yet been reviewed and accepted by the "official channel," such as a working group consisting of representatives from the project team, product management, and user or customer community.]*
Approved	*[Capabilities that are deemed useful and feasible, and have been approved for implementation by the official channel.]*
Incorporated	*[Features incorporated into the product baseline at a specific point in time.]*

[If the amount of incoming requirements is large, another set of status attributes could be the one suggested in the REPEAT Lifecycle (Regnell)]

New	*A requirement is elevated to the assigned state when an expert ahs been assigned to investigate the requirement and determine the value of a number of attributes.*
Assigned	*When reaching this state, an expert has assigned values to attributes representing a rough estimate of cost and architectural impact. Comments and implementation ideas may also be stated.*
Classified	*All requirements in this state are selected for implementation for the coming release. They are sorted in priority order on two list: a must-list for mandatory requirements and a wish list for "nice-to-have" requirements. They also have attributes assigned concerning detailed cost and impact estimations. There is also a more detailed textual specification of the requirement. A selected requirement may be de-selected, due to changed circumstances, and the re-enters the classification state or gets rejected.*
Selected	*This is an end-state indicating that the requirement has been implemented and verified. The requirement is now incorporated in a release that can be marketed to customers.*
Applied	*This is an end-state indicating that the requirement has been rejected, e.g. because it is a duplicate, already implemented, or It does not comply with the long-term product strategy.*
Rejected	

A.3 Benefit

[Set by Marketing, the product manager or the business analyst. All requirements are not created equal. Ranking requirements by their relative benefit to the end user opens a dialogue with customers, analysts, and members of the development team. Used in managing scope and determining development priority.]

Critical	*[Essential features. Failure to implement means the system will not meet customer needs. All critical features must be implemented in the release or the schedule will slip.]*
Important	*[Features important to the effectiveness and efficiency of the system for most applications. The functionality cannot be easily provided in some other way. Lack of inclusion of an important feature may affect customer or user satisfaction, or even revenue, but release will not be delayed due to lack of any important feature.]*
Useful	*[Features that are useful in less typical applications will be used less frequently or for which reasonably efficient workarounds can be achieved. No significant revenue or customer satisfaction impact can be expected if such an item is not included in a release.]*

[An alternative set of benefit attributes is one that is focuses on the reaction of the customer. This set is collected from Cohen, 1995.]

Sensational requirements	*[The customers will be impressed if the product includes this requirement. If not included the customer will not take any notice, since it fulfills a need the customer did not know he or she had.]*
Normal requirements	*[Requirements expressed by customers. If well implemented, the customer degree of satisfaction is higher.]*
Expected requirements	*[The customer is disappointed if this requirement is not met. However, if included, the degree of satisfaction will not get any higher.]*

A.4 Effort

[Set by the development team. Because some features require more time and resources than others, estimating the number of team or person-weeks, lines of code required or function points, for example, is the best way to gauge complexity and set expectations of what can and cannot be accomplished in a given time frame. Used in managing scope and determining development priority.]

A.5 Risk

[Set by development team based on the probability the project will experience undesirable events, such as cost overruns, schedule delays or even cancellation. Most project managers find categorizing risks as high, medium, and low is sufficient, although finer gradations are possible. Risk can often be indirectly assessed by measuring the uncertainty (range) of the projects team's schedule estimate.]

A.6 Stability

[Set by analyst and development team based on the probability the feature will change or the team's understanding of the feature will change. Used to help establish development priorities and determine those items for which additional elicitation is the appropriate next action.]

A.7 Assigned To

[In many projects, features will be assigned to "feature teams" responsible for further elicitation, writing the software requirements, and implementation. This simple pull-down list will help everyone on the project team to understand responsibilities better.]

A.8 Dependencies

[Dependencies can be of different types, each affecting how valuable or complicated a certain feature will be to implement. Having a clear picture of how requirements depend on each other will make it easier to make good decisions while planning the release. The examples of dependency attributes are collected from Carlshamre, A Usability Perspective on Requirements Engineering, 2001.]

AND	*A printer requires a driver to function, and the driver requires a printer to function.*
REQUIRES	*Sending an e-mail requires a network connection, but not the opposite.*
TEMPORAL	*The function* Add object *should be implemented before* Delete object.
CVALUE	*A detailed on-line manual may decrease the customer value of a printed manual.*
ICOST	*A requirement stating that "no response time should be longer than 1 second" will typically increase the cost of implementing many other requirements.*

<Project Name>
Business Case

Version <1.0>

[Note: The following template is provided for use with the Rational Unified Process. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

Revision History

Date	Version	Description	Author
<dd/mmm/yy>	<x.x>	<details>	<name>

Table of Contents

Business Case

1. Introduction

*[The introduction of the **Business Case** should provide an overview of the entire document. It should include the purpose, scope, definitions, acronyms, abbreviations, references, and overview of this **Business Case**.]*

1.1 Purpose

*[Specify the purpose of this **Business Case.**]*

1.2 Scope

*[A brief description of the scope of this **Business Case**; what Product(s) it is associated with, and anything else that is affected or influenced by this document.]*

1.3 Definitions, Acronyms and Abbreviations

*[This subsection should provide the definitions of all terms, acronyms, and abbreviations required to properly interpret the **Business Case**. This information may be provided by reference to the project Glossary.]*

1.4 References

*[This subsection should provide a complete list of all documents referenced elsewhere in the **Business Case**. Each document should be identified by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]*

1.5 Overview

*[This subsection should describe what the rest of the **Business Case** contains and explain how the document is organized.]*

2. Product Description

*[To give a context to the reader, briefly describe the product that is to be developed. Include the name of the system and possibly an acronym, if one is used. Explain what problem it solves and why the development will be worth the effort. Refer to the **Vision** document.]*

3. Business Context

[Define the business context for the product. In which domain is it going to function (for example, telecom or bank) and what market—who are the users? If it is a continuation of an existing product, this should also be mentioned.]

4. Product Objectives and Roadmap

[State the objectives for developing the product— the reasons why this is worthwhile. This includes a tentative schedule, and some assessment of schedule risks. Clearly defined and expressed objectives provide good grounds for formulating milestones and managing risks; that is, keeping the project on track and ensuring its success. Also include any plans for subcontracting and/or use of COTS and clarify why that is an advantage.]

5. Financial Forecast

[For a commercial software product, the Business Case should include a set of assumptions about the project and the result of a profitability analysis based on those assumptions. For example, the ROI will be a magnitude of five if completed in one year, two if completed in two years, and a negative number after that. These assumptions are checked again at the end of the elaboration phase when the scope and plan are known with more accuracy. The return is based on the cost estimate and the potential revenue estimates. Make sure to include service, support, distribution and marketing costs.

The resource estimate encompasses the entire project, through to delivery. This estimate is updated at each phase and each iteration, and becomes more accurate as each iteration is completed.

An explanation of the basis of estimates should be included.]

6. Constraints

[Express the constraints under which the project is undertaken. These constraints impact risk and cost. They could be things like external interfaces that the system must adhere to, standards, certifications or a technical approach employed for strategic reasons, such as using a certain database technology or distribution mechanisms.]

7. Funding

[Explain the financing plan for the project.]