Tillämpning av prototyper i Rational Unified Process

Petros Leventis & Ann-Christin Tomazic

Handelshögskolan vid Göteborgs universitet
Institutionen för informatik

Abstrakt

Uppsatsens syfte var att utforska dagens prototyputveckling, systemutvecklingsmetoden RUP (Rational Unified Process) samt hur prototyputveckling fungerar inom RUP. Syftet uppnådes med hjälp av det hermeneutiska förhållningssättet tillsammans med den kvalitativa metoden enligt lineär disposition.
I litteraturstudierna definierades prototyputvecklingens syften, utvecklingsmetoder och tekniker samt RUP:s systemutvecklingsteori, struktur och dess förhållande med prototyputveckling.
I de empiriska studierna, främst genom intervjuer, undersöktes hur prototyputveckling respektive RUP fungerar i praktiken samt hur de fungerar tillsammans.
Undersökningarna utfördes i ett enda web-projekt inom Web Program Center på Volvo Information Technology AB. Därav framkom att behov av prototyputveckling fanns bland utvecklarna. Den rådande begreppsförvirringen som fanns inom prototyputveckling, försvårade beslutfattandet kring prototypernas utveckling inom projektet.
RUP visade sig vara ett bra stöd för systemutveckling, men p.g.a. dess komplexitet borde man anpassa RUP till varje projekt. Även utvecklarnas ovana vid RUP borde tas hänsyn till vid planering av projekten. Slutligen visade uppsatsen att prototyputveckling inte påverkas av RUP, men RUP i sin tur använde sig av prototyper för att på ett effektivt sätt stödja systemutvecklingsarbetet.

Magisteruppsats 20p
Höstterminen 2001

Handledare: Göran Walske
Examinator: Thanos Magoulas

Förord

För att kunna skriva en uppsats som denna har vi varit beroende av ett antal personers insatser.

Först och främst vill vi tacka Christina Rux, vår handledare på WPC, för att hon frågade om vi ville skriva vårt arbete hos dem och för all inspiration, alla idéer och feedback som vi har fått.

Vi vill tacka utvecklarna på avdelningen ”Solution Development” på WPC, som ställde upp på intervjuer trots att de var mitt inne i en väldigt hektisk period av projektet, samt alla andra inom WPC för all uppmuntran och för att vi fick en arbetsplats hos er.

Ett stort tack vill vi ge till Runo Burman från avdelningen Method and Tecniques på Volvo IT samt Charlie Nordblom från AB Volvo, för oerhört intressanta och givande intervjuer inte bara för vår uppsats skull utan även för ökandet av vår egen förståelse.
Utan er och WPC:s input hade det blivit en mycket tunn uppsats.

Ett stort tack vill vi även ge Jessika Andersson & Ulrika Johansson på avdelningen Method and Techniques på Volvo IT som gav oss grundliga genomgångar av RUP och som även under arbetets gång har svarat på våra frågor om RUP.

Men framför allt vill vi tacka vår handledare på institutionen för informatik, Göran Walske, för all uppmuntran, engagemang och entusiasm. Vi körde fast ett antal gånger, men efter varje handledarträff med Göran kom vi tillbaka till vår arbetsplats glada och fyllda med inspiration och nya idéer.

Sist men inte minst, vi vill tacka våra respektive sambos Maria och Dan för stöd och uppmuntran under detta arbete.

Göteborg januari 2002

Petros Leventis & Ann-Christin Tomazic

Innehållsförteckning

1 Introduktion

1.1 Inledning

Under vår utbildning på institutionen för Informatik har det både i litteratur och på föreläsningar framhållits att prototyputveckling är en viktig del av systemutvecklingsarbetet[1].
Den främsta anledningen till att man skapar prototyper påstås vara att man vill ta fram eller förtydliga en kravspecifikation, testa användargränssnitt[2] och användbarhet samt för att upptäcka risker tidigt i systemutvecklingsprocessen vilket leder till reducerade systemutvecklingkostnader. (Sommerville, 2001)
Vi uppfattade prototyputveckling som något positivt, men vi funderade på om prototyper verkligen är något man använder sig av ute på företagen och om det är relevant och ekonomiskt försvarbart för ett företag att utveckla prototyper?

Många betraktar en prototyp som en engångsprototyp, d.v.s. en prototyp som utvecklas till det kompletta informationssystemet och som kastas efter utvärdering. Därefter utvecklas det ”riktiga” systemet utifrån de krav och riktlinjer man får fram av prototypen. Andra hävdar att papper och penna kan räcka för att utveckla en prototyp.
(Preece 1994; Sommerville, 2001; Löwgren, 1993)

Med ovanstående i åtanke insåg vi att begreppet *prototyp* inte har en exakt definition inom systemutveckling. Denna förvirring kring begreppet bland systemutvecklare och andra intressenter försvårar beslutsfattandet kring prototyputveckling i och med att prototyper uppfattas olika av olika personer.
Vi har själva haft svårt att få en klar bild av vad prototyputveckling egentligen är och hur den fungerar i praktiken, trots all litteratur vi har läst och alla föreläsningar vi har fått.

Inom systemutveckling i allmänhet finns ett behov av standardiseringar av olika slag, däribland även för systemutvecklingsmetoder. Många företag utvecklar egna systemutvecklingsmetoder för att de ska passa den egna verksamheten. Höga utbildningskostnader för nyanställd personal samt höga utvecklings- och underhållskostnader av den egna systemutvecklingsmetoden leder till behov av en standard.

Rational Software har tagit tillfället i akt att utveckla en systemutvecklingsmetod, RUP (Rational Unified Process) som stöds av olika verktyg som t.ex. Rational Rose. Detta ”paket” marknadsför de och säljer gärna som den nya blivande standarden.
(Kruchten, 2000)

Rapporter och litteratur om prototyputveckling har hittills baserats på tidigare systemutvecklingsmetoder. Då RUP är en relativt omdiskuterad systemutvecklingsmetod, som har fått en snabb spridning inom branschen, såg vi möjlighet att undersöka hur RUP egentligen uppfattas bland utvecklare samt hur prototyputveckling fungerar inom denna systemutvecklingsmetod.

---

[1] Med systemutveckling avses utveckling av informationssystem.
[2] Med användargränssnitt avses **grafiska** användargränssnitt

WPC (Web Program Center) är en av avdelningarna på Volvo IT (Volvo Information Technology AB) som sedan en tid tillbaka tillämpar RUP för att utveckla sina system, vilket gör det till en passande miljö för våra undersökningar.

1.2 Frå geställning

Med vår rapport vill vi undersöka den prototyputveckling som utförs idag för att se *om* och *hur* teorin fungerar i praktiken och då tillsammans med RUP.

De frågor som har väckt vårt intresse och som vi ämnar besvara i denna uppsats är följande:

- Vilken är den allmänna uppfattningen om prototyputveckling hos systemutvecklarna på WPC, på avdelningen Methods and Techniques inom Volvo IT och hos kunden?
- Vilka tydliga för- och nackdelar finns det med prototyputveckling?
- Vilken är den allmänna uppfattningen om RUP hos systemutvecklarna inom WPC, på avdelningen Methods and Techniques inom Volvo IT och hos kunden?
- Vilka prototyputvecklingsmetoder används och när blir de aktuella i RUP?

1.3 Avgränsning

Man behöver inte vara systemutvecklingsexpert för få behållning av vår uppsats, men det underlättar om man har grundläggande kunskaper inom ämnet.

Vårt intresseområde är stort och brett och samtidigt mycket intressant. Det finns mycket vi skulle vilja fördjupa oss i, men tiden är begränsad varför vi helt enkelt är tvungna att göra vissa avgränsningar.

- Vi har valt att inrikta oss på ett enda projekt inom WPC, som de flesta inom avdelningen för tillfället arbetar med. Vår undersökning gäller således ett enda web-projekt på Volvo IT.

- Frågan om det är ekonomiskt försvarbart för företag att utveckla eller låta bli att utveckla prototyper tycker vi är en viktig och intressant fråga, men vi har inte inriktat oss på de ekonomiska aspekterna av prototyputveckling. Våra kunskaper inom företagsekonomi är helt enkelt för otillräckliga för att genomföra sådana ekonomiska analyser.

- Litteraturen nämner mycket om olika verktyg som kan användas vid prototyputveckling, men vi går inte närmare in på dem p.g.a. att vi vill försöka hålla oss på en teoretisk nivå.

- Vi fördjupar oss inte heller i Rationals olika verktyg och inte heller i UML (Unified Modeling Language) som är den standardnotation som Rationals verktyg använder sig av.

1.4 Disposition

Kapitel 2 - Vetenskaplig metod - beskriver hur vi ska gå tillväga för att besvara vår frågeställning.

Kapitel 3 - Definition av prototyputveckling - baseras på litteraturstudier och vi ger där ger vår tolkning av prototyper och prototyputveckling.

Kapitel 4 – Definition av RUP - baseras på litteratur och onlinedokumentation om RUP. Kapitlet ger en grundläggande sammanfattning av RUP:s grundbegrepp och uppbyggnad.

Kapitel 5 - Intervjuresultat - presenterar tillvägagångssätt, tolkning och resultat av våra genomförda intervjuer.

Kapitel 6 - Diskussion och slutsatser - presenterar våra diskussioner och slutsatser baserade på de genomförda intervjuerna.

Kapitel 7 - Förslag kring prototyputveckling, RUP samt prototyputveckling i samband med RUP – ger vi våra förslag på prototyputveckling, RUP samt prototyputveckling i samband RUP.

Kapitel 8 - Förslag till framtida forskning - ger förslag på forskningsområden till andra uppsatsskrivare.

Till sist följer referenser och bilagor.

2 Vetenskaplig metod

Hermeneutiskt förhållningssätt

Vi använder ett hermeneutiskt förhållningssätt för att utöka vår grundläggande förståelse av det område vi valt studera och undersöka.
Ett hermeneutiskt förhållningssätt handlar om att tolka och analysera de erfarenheter som man samlat in, så att man härigenom skaffar sig kunskap och förståelse för en människas värderingar och bakgrunden eller orsaken till hennes handlingar. Ett positivistiskt förhållningssätt däremot, har som syfte att, via konkreta observationer och kontrollerade experiment, fastslå objektivt säkra fakta. (Starrin, 1994)

Kvalitativ metod

I hermeneutisk forskning kan man använda sig av kvalitativ respektive kvantitativ metod, i kombination eller var för sig. I vår uppsats har vi valt att tillämpa den kvalitativa metoden som kännetecknas av verbala formuleringar. Individens uppfattning och tolkning av sin verklighet står i centrum. Den kvantitativa metoden bygger däremot i huvudsak på matematiska och statistiska mätningar.

I vår uppsats använder vi oss av lineär disposition dvs. introduktion, problem, vetenskaplig metod, resultat och diskussion. Det är en i forskningssammanhang mycket vanlig struktur. (Backman, 1998)

Litteraturstudier

Inför litteraturstudierna försökte vi hitta litteratur som enbart var inriktade på prototyputveckling. Vi trodde att det skulle finnas gott om litteratur i ämnet, eftersom det poängteras som något viktigt inom systemutveckling. Men vi hittade inte några böcker som handlar endast om prototyputveckling. Därför fick det istället bli böcker som handlar om systemutveckling samt mjukvaruutveckling och där vanligtvis ett, eller en del av ett kapitel ägnas åt prototyputveckling.

Vi har även använt oss av tidningsartiklar och vi har sökt efter information på Internet.

När det gäller litteratur om RUP är utbudet även här begränsat. Som vi har nämnt tidigare, antar vi att det beror på att RUP är en ny företeelse inom systemutveckling. Således har vi baserat vår litteraturstudie om RUP på Rationals eget material, en bok samt RUP online, som är Rationals introduktion till RUP på CD-rom.
Vi hoppades att vi skulle hitta litteratur om hur prototyputveckling fungerar inom RUP, men vi hittade ingen litteratur just om detta.

Empiriska studien

Litteraturstudier inom prototyputveckling och RUP har varit en betydelsefull del i vårt arbete. För att intervjuresultaten skulle bli så bra som möjligt, var det viktigt att vi själva hade en klar uppfattning om och stor förståelse av dessa två områden, allt enligt det hermeneutiska förhållningssätt som vi har valt i vår uppsats.
Slutligen ansåg vi att vi var tillräckligt förberedda för att kunna formulera lämpliga intervjufrågor inför den empiriska studien.
Då vi har valt den kvalitativa metoden för vår uppsats blev det ganska självklart att den empiriska delen skulle bestå av intervjuer.

Vi genomförde ett personligt samtal med var och en av projektintressenterna, där de med egna ord redogjorde för sin uppfattning om prototyputveckling, RUP samt deras inbördes förhållande. Den främsta fördelen med intervjuförfarandet var att vi hade möjligheten att ställa motfrågor till de intervjuade, vilket hade varit omöjligt i t.ex. en kvantitativ enkätundersökning. Slutligen hade vi, genom litteraturstudien tillsammans med den empiriska studien, fått ett omfattande material att använda i våra diskussioner och till att dra slutsatser kring vår frågeställning.

3 Definition av Prototyputveckling

3.1 Prototyper inom systemutveckling

Vi har tagit Nationalencyklopediens ordbok till hjälp för att få en definition av ordet prototyp. Prototyp betyder *”första exemplar av något som t.ex. utgör en mall vid serietillverkning*”. Det kommer av grekiskans ”prototypon” som betyder ”original” där ”protos” betyder ”först, ursprunglig”. (NE, 1989)
Inom systemutveckling har begreppet ”prototyp” ärvts från den traditionella industrin, men innebörden är inte riktigt densamma. Utvecklingen av den industriella prototypen kännetecknas av att den sker under en lång tidsperiod, att kostnaderna för prototypen är större än för den slutliga produkten, samt att det av prototypen skapas en produkt som börjar serietillverkas. (Stevens, 1998)

Tabellen (tabell 3-1) visar förhållandet/skillnaden mellan industriell prototyputveckling och prototyputveckling inom systemutveckling.

Tabell 3-1: Prototyputveckling inom olika industrier (Stevens, 1998, s. 249)

Industri	Antal exemplar	Designkostn.	Enhetskostn.	Prototypens kännetecken
Bil	100.000 - 10.000.000	Mycket höga	Höga	Stor insats vid framtagning av prototyp. Stabila krav. Serietillverkning
Mjukvara	1 - 5.000.000	Höga	Nära noll	Ständigt varierande användar- och funktionalitetskrav. Ingen tillverkningsprocess. Hög felfrekvens i praktiken.

I facklitteratur och i tidningsartiklar påpekas ofta att prototyputveckling[3] i samband med systemutveckling leder till ett bättre fungerande system till en reducerad kostnad. (Sommerville, 2000)

Men vad menas egentligen med en prototyp inom systemutveckling?
I artikeln ”Rätt ord för bransch och industri” skriver Anders Lotsson (2001) om förvirringen mellan olika begrepp, bl.a. om ordet prototyp.
Hans åsikt är att ”*en prototyp är en fungerade provisorisk version av en produkt …*” som ”*byggs för teknisk testning, användartestning och demonstration*”. De icke funktionella prototyperna bör inte kallas för prototyper, enligt Lotsson, utan borde kallas attrapp eller modell.

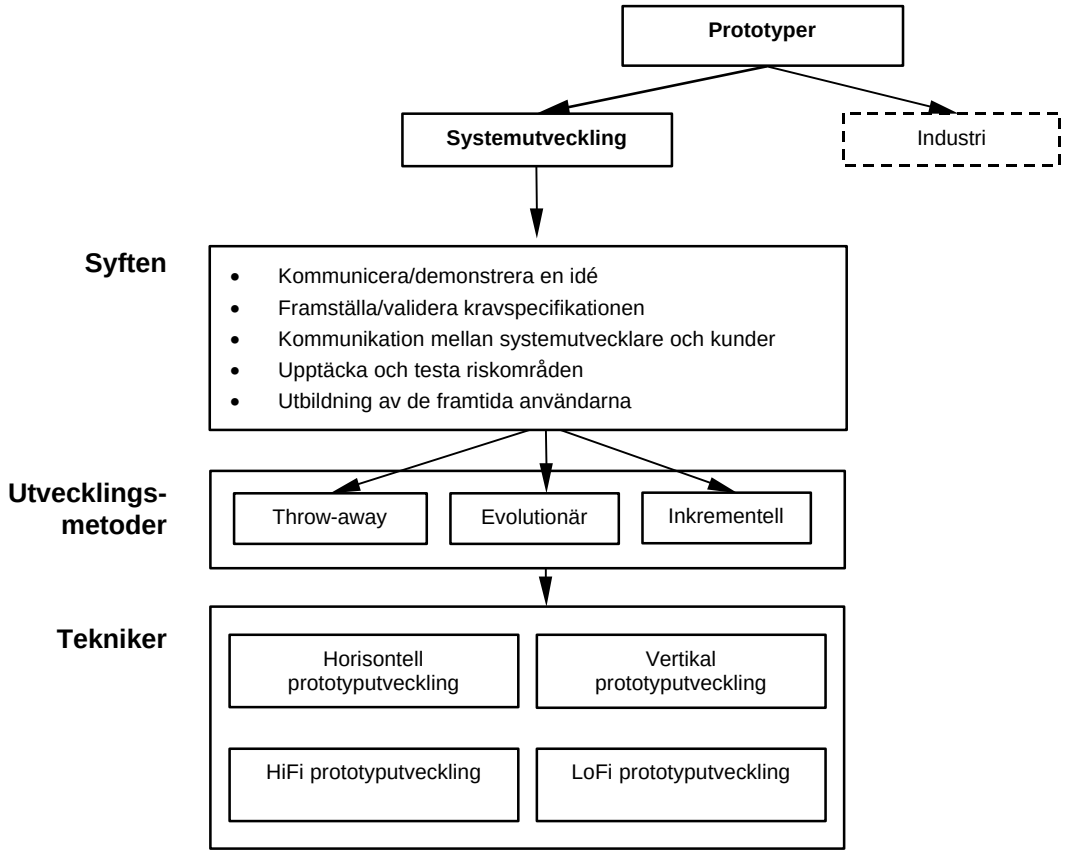
Mathiassen, Munk-Madsen, Nielsen och Stage (1998) har en annan aspekt på begreppet prototyp. De anser att det räcker att en prototyp består av skissade skärmbilder och fönster, funktioner som bara accepterar vissa bestämda indata och att informationslagringen är helt statisk. Löwgren (1993) menar emellertid att en prototyp visst inte behöver vara en programvara. I vissa fall kan det räcka med t.ex. papper och tejp, PowerPoint-bild eller något liknande.

[3] I fortsättningen avser vi prototyputveckling inom systemutveckling när vi nämner prototyputveckling.

Alla nämnda författare har försökt att formulera sina tankar och presentera en egen förklaring till vad prototyputveckling bör vara. Genom deras beskrivningar har vi kunnat identifiera tre olika grundelement - syfte, utvecklingsmetod och tekniker - vilka enligt vår uppfattning tillsammans ger en ännu mer konkret bild av prototyputveckling.

Prototyputveckling har alltid ett syfte. Beroende på syfte väljer man en utvecklingsmetod som i sin tur använder sig av en eller flera tekniker.

Figur 3-1 visar vår bild av samspelet mellan dessa tre grundelement. I resten av detta kapitel presenterar vi mer detaljerat tänkandet bakom varje grundelement och dess byggstenar.



Figur 3-1: Vår bild av samspelet mellan de tre grundelementen.

3.1.1 Vad är syftet med prototyper?

Vad är då det egentliga syftet med prototyputveckling?
Mathiassen et al. (1998, s. 52) menar att: ”*Avsikten med ett experiment med en prototyp är alltid att ta reda på någonting som man inte vet*.”
Under en systemutvecklingsprocess brukar det dyka upp många oklarheter och obesvarade frågor för systemutvecklarna, t.ex. vad vill användarna egentligen att det nya systemet ska kunna göra. Mathiassen et al. (1998) hävdar att utveckla en prototyp är ett bra sätt för att ta reda på oklarheter och obesvarade frågor.

Beroende på vad man vill använda prototypen till kan man ha olika syften med prototyputveckling.
Syftena kan delas upp i följande fem huvudgrupper:

- Kommunicera/demonstrera en idé.
- Framställa/validera kravspecifikationen .
- Kommunikation mellan systemutvecklare och kunder.
- Upptäcka och testa riskområden.
- Utbildning av de framtida användarna.

(Löwgren, 1993; Preece, 1994; Sommerville, 2001)

Vi vet alla hur svårt det kan vara att försöka förklara en komplex idé för någon annan person utan att ha något konkret exempel att visa.
Löwgren uttrycker det: ”*An idea is hard to evaluate; a prototype can be evaluated in several different ways*.” (Löwgren, 1993, s. 99)
Med hjälp av en prototyp i en sådan situation kan man i förväg *demonstrera sin idé* utan att lägga ner alltför mycket resurser, tid och pengar. (Löwgren, 1993)
Prototypen skapar en konkret bild av problemområdet[4] och underlättar bedömningen av idén. (Sommerville, 2001)
En prototyp kan också fungera som ett underlag för beslutsfattande. I ett projekt kan man ta fram flera olika prototyper av problemområdet vilka utvärderas var för sig och därefter kan man besluta om vilken variant man vill fortsätta att arbeta med. (Löwgren, 1993)

En av de främsta anledningarna till att utveckla prototyper är att *ta fram en kravspecifikation* för det framtida systemet och även här är *kommunikationen mellan systemutvecklare och kunder* en viktigt faktor. I de flesta fall kan det vara svårt för kunderna att specificera sina krav på systemet, d.v.s. kunna uttrycka vad de vill att det nya systemet ska kunna göra, och veta hur systemet kommer att interagera med andra system. Med en prototyp kan slutanvändarna prova på hur det nya systemet kan komma att stödja deras arbete och hitta svagheter i det framtida systemet. Prototypen hjälper dem att kommunicera sina nyupptäckta krav och nya idéer till systemutvecklarna. Fördelen är att systemutvecklarna får kravspecifikationen validerad och det leder till att de bygger ”rätt system från början”, d.v.s. att de bygger ett system som kunderna kan och vill använda. (Sommerville, 2001)

Ett annat syfte med prototypen är att *upptäcka och testa olika riskområden*. Den har ofta till uppgift att simulera en viss del av det framtida systemet som av systemutvecklarna har bedömts som riskabel. Ett exempel är ett system som ska interagera med andra redan befintliga system. Genom att man simulerar ett sådant samspel mellan olika system, kan man testa möjligheten för samarbete mellan dem. Efter testerna, beroende på resultat, kan kravspecifikationen anpassas till de nya förutsättningarna. Om t.ex. ett test har visat att interaktion med ett visst system är omöjlig, alldeles för riskabel eller för komplicerad ska kravspecifikationen omförhandlas så att den passar den nya situationen. (Sommerville, 2001)

När prototypen väl finns tillgänglig kan man passa på att utnyttja den i *utbildningssyfte* och ge användarna en möjlighet att öva sig innan det slutliga systemet levereras.

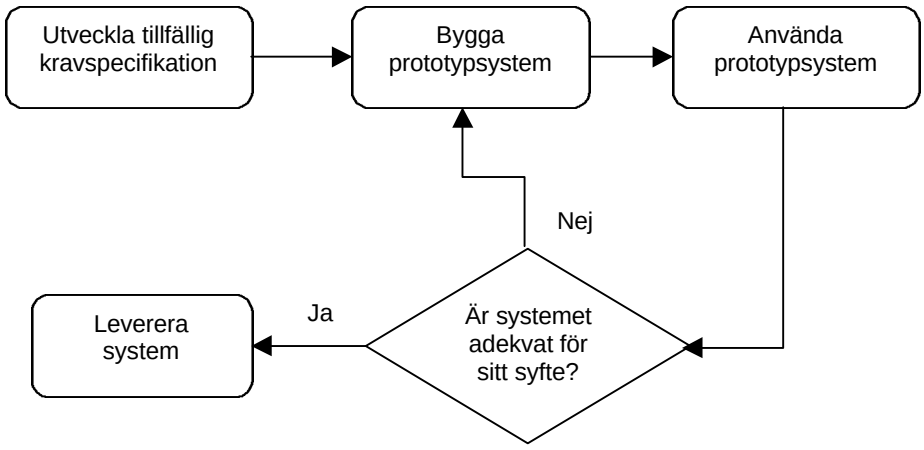
[4] Problemområde enligt Mathiassen et al. (1998) är ” den del av omgivningen som administreras, övervakas eller styrs med hjälp av ett datasystem”.

3.1.2 Utvecklingsmetoder för prototyputveckling

3.1.2.1 Evolutionär prototyputveckling

Som tidigare har nämnts är användarna en viktig del av systemutvecklingsprocessen. Det är ju användarna som slutligen ska använda det nya systemet i sitt dagliga arbete och då är det viktigt att systemet motsvarar de krav användarna har på systemet för att med dess hjälp kunna lösa sina arbetsuppgifter.
Preece (1994), Sommerville (2001), Stevens (1998) m.fl. nämner den evolutionära prototypmetoden som ett sätt att skapa ett system där användarna kontinuerligt är med och påverkar utvecklingen av sitt system.
Då det kan vara svårt för användarna att hitta och specificera alla sina krav utgår den evolutionära utvecklingsmetoden ifrån att man bygger en prototyp utifrån de viktigaste delarna i kravspecifikationen, d.v.s. de delar som är enklast att förstå (tydligast uttryckta) och som kunden anser sig ha störst behov av. Otydliga krav och krav med lägre prioritering införs i prototypen först när och om användarna kräver det.

Den evolutionära utvecklingsmetoden är en iterativ process. Genom att låta användarna använda det ofullständiga systemet upptäcker de snart nya krav som läggs till i prototypen. Prototypen utvecklas allteftersom användarnas krav läggs till och slutligen har ett fungerande system levererats till användarna. (Se figur 3-3).



Figur 3-1: Evolutionär prototyputveckling (Sommerville, 1991, s. 176).

Fördelarna med evolutionär prototyputveckling är att systemet levereras snabbt till kunden och att användarna deltar i systemutvecklingsarbetet på ett mer aktivt sätt. Snabb leverans innebär lägre kostnader för projektet och i och med användarnas medverkan blir upplärningstiden vid införandet av det nya systemet kortare. (Sommerville, 2001)

De främsta nackdelarna med denna prototyputvecklingsmetod gäller verifiering (att i *simulerad* miljö finna fel hos en programvara, m.a.o. ”bygger vi systemet rätt?”) och validering (att i *verklig* miljö finna fel hos en programvara, m.a.o. ”bygger vi rätt system?”).
Verifiering går ej att genomföra eftersom det inte finns någon systemspecifikation från början, denna tas ju fram allt eftersom systemet växer fram.

Validering går inte heller att genomföra fullt ut eftersom vi ej heller här har tillgång till en ursprunglig systemspecifikation. Det enda vi kan testa är att systemet är adekvat för sitt syfte.

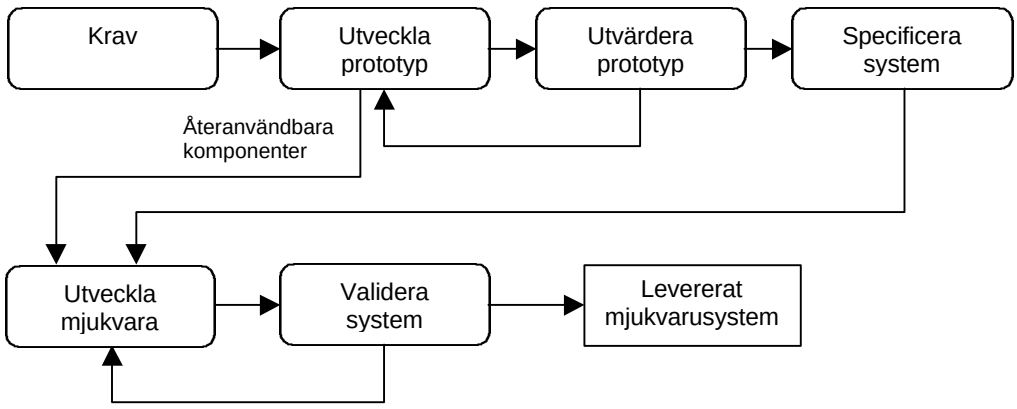
En annan nackdel, enligt Preece (1994), är att den evolutionära prototypen uppmuntrar systemutvecklarna att ”fixa till” nytillkomna krav istället för att utforska ytterligare alternativ till lösningar på problemen. Systemutvecklarna ändrar ogärna i sina lösningar utan väljer istället att lösa problemet för stunden. Vad man kanske inte tänker på är att dessa ”fixa till-lösningar” kan leda till större problem i ett senare skede av systemutvecklingsprocessen.

Von Dorrien (2001) menar att då evolutionär prototyputveckling innebär ständiga iterationer och att nya krav tillkommer även mot slutet av systemutvecklingsprocessen blir det till slut dyrare och svårare att åtgärda dessa problem. Därför kan man säga att evolutionär prototyputveckling passar bäst till utveckling av små system och delar av system, då dessa typer av projekt oftast inte är så långa och oftast inte är så kostsamma.

3.1.2.2 Throw-away prototyputveckling

När det gäller throw-awayutvecklingsmetoden har man redan från början en mycket tydligare kravspecifikation. Man väljer här att utveckla prototypen för de delar som är mest vagt uttryckta i kravspecifikationen, detta för att få fram mer detaljerad information till kravspecifikationen. De delar i kravspecifikationen som är tydliga behöver kanske aldrig utvecklas/modelleras i throw-awayprototypen.

Throw-away prototypen har så kort livstid att man kortfattat kan säga att throw-awayprototypen utvecklas, utvärderas och modifieras. Utifrån prototypen skapar man sedan en systemspecifikation. När man har lyckats åstadkomma en fullgod systemspecifikation kastas prototypen och systemet fortsätts att utvecklas utifrån den kompletterade systemspecifikationen. (Se figur 3-4)



Figur 3-1: Throw-away prototyputveckling (Sommerville, 1991, s. 179).

Throw-awayprototypens uppbyggnad skiljer sig från det slutliga systemet. Tanken med throw-awayprototypen är att den ska vara billig och ska gå snabbt att ta fram.
Den har t.ex. inte krav på sig att vara fullt funktionell eller ha hög prestanda, vilket leder till att man kan utveckla och utvärdera flera olika prototypförslag samtidigt.

Programmeringsspråket i prototypen är oftast inte heller det språk man kommer att använda i det slutliga systemet. (Sommerville, 2001)

En viktig fördel med throw-awayprototypen, om man jämför med den evolutionära, är att man kan validera och verifiera prototypen eftersom man har en system-specifikation att jämföra med. (Sommerville, 2001)

I vissa fall, t.ex. om prototypen är skriven i samma språk som det slutliga systemet, kan det vara kostnadseffektivt att återanvända komponenter av prototypen, men det är förknippat med vissa risker. T.ex kan det handla om ineffektiv kod, som kan ge systemet sämre prestanda.

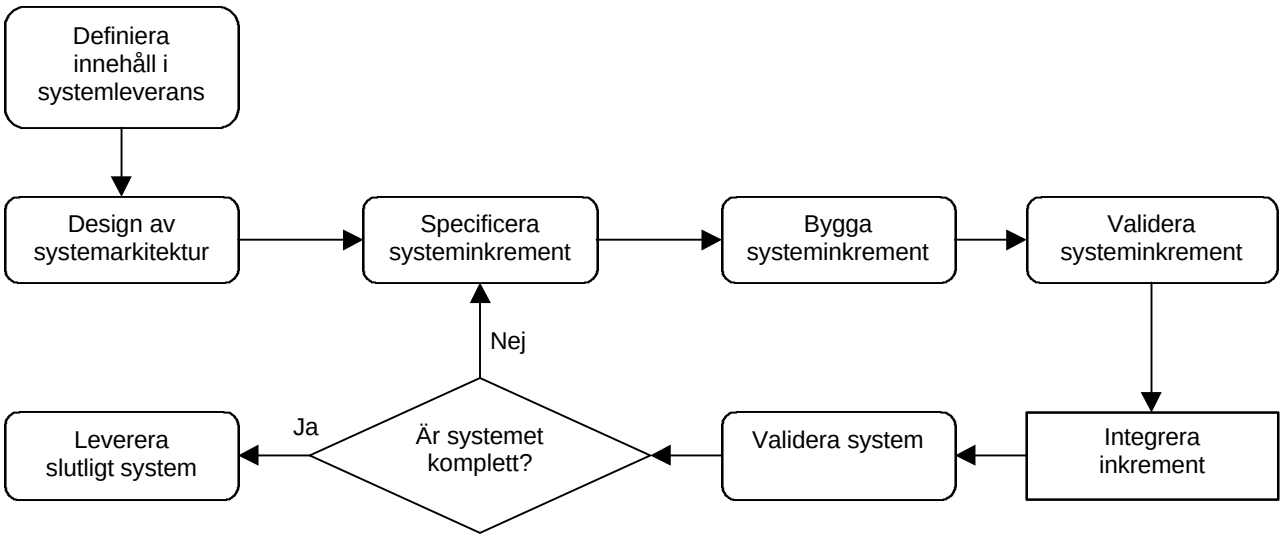
Nackdelen med throw-awayprototypen är att de enkla prototyper man tar fram har vissa kvalitetsbrister. Risken är att testpersonerna bortser från dessa brister vid ett test och istället löser testuppgifterna på annat sätt, vilket i sin tur kan ge felaktiga testdata. (Sommerville, 2001)

Ibland trycker kunden på att throw-awayprototypen skall levereras som det färdiga systemet, speciellt när leveranstider blir svårare att hålla, men enligt Sommerville (2001) är det inte att rekommendera, då:

- det kan vara svårt att anpassa prototypen till verkliga krav beträffande robusthet, prestanda, pålitlighet mm.
- snabba ändringar under prototyputvecklingen innebär att dokumentationen blir bristfällig. Det enda man har är prototypkoden, vilken inte är tillräcklig för att möjliggöra framtida underhåll.
- det kan komma att bli dyrt och svårt att underhålla systemet eftersom systemstrukturen inte har prioriterats under prototyputvecklingen.
- organisationens kvalitetskrav har eftersatts när det gäller prototypen, vilket innebär att prototypen inte håller samma kvalitet som ett fullt utvecklat system.

3.1.2.3 Inkrementell prototyputveckling

Inkrementell prototyputveckling undviker problemen med ständiga kravförändringar som karakteriserar evolutionär prototyputveckling.
Istället upprättas en övergripande systemarkitektur, ett ramverk, i början av processen. Systemkomponenterna utvecklas inkrementellt, d.v.s. stegvis, och levereras inom detta ramverk. När dessa har utvärderats och levererats ändras varken ramverk eller komponenter såvida inte ytterligare fel upptäcks. Eventuellt kan feedback från användare påverka designen av komponenter i ett senare skede.
Inkrementell prototyputveckling är dessutom mer hanterbar, då planer och dokumentation måste produceras inför varje systeminkrement (Sommerville, 2001)



Figur 3-1: Inkrementell prototyputveckling (Sommerville, 1991, sid 178).

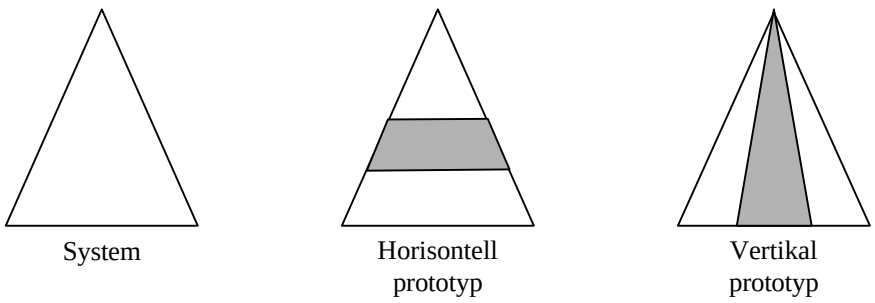
3.1.3 Tekniker för prototyputveckling

3.1.3.1 Horisontell prototyputveckling

Horisontell prototyputveckling innebär att man bygger en prototyp på en utvald ”del” för hela systemet, t.ex. användargränssnittet.
Prototypen kan vara en s.k. attrapp, d.v.s. den behöver inte vara funktionell.
Den horisontella prototypen är bäst lämpad för presentation av gränssnitt mot användare och andra system. (Preece, 1994; Stevens, 1998)

3.1.3.2 Vertikal prototyputveckling

Vertikal prototyputveckling utforskar inom ett riskfyllt område en mindre sektion av systemets alla delar i detalj. Resultatet visar all funktionalitet för den sektionen genom hela systemet.
Ett exempel kan vara ett kommunikationsprotokoll som systemet skall använda mot andra system (Preece, 1994; Stevens, 1998)



Figur 3-1: Horisontell och vertikal prototyputveckling (Stevens, 1998, s. 251).

3.1.3.3 High-Fidelity prototyputveckling (HiFi)

HiFi-prototypen använder sig av teknik för att i möjligaste mån likna den slutliga produkten, både när det gäller teknologi och arkitektur, vilket också leder till att HiFi-prototypen ger mest tillförlitliga testdata.
När prototypen motsvarar de ställda kraven kan den införas, med få eller inga ändringar, i det slutliga systemet.
HiFi-prototypen är populär i marknadsföringssammanhang eftersom den ger en första visning av den kommande produkten och dessutom ser den estetisk och ”välpolerad” ut.
(Preece, 1994; Löwgren, 1993)

3.1.3.4 Low-Fidelity prototyputveckling (LoFi)

LoFi-prototypen skapas med enklare hjälpmedel, t.ex. PowerPoint, olika animerade bilder och ibland post-itlappar. Fördelen är att tekniken är billig och snabb att ta fram, vilket gör att många olika förslag kan utforskas samtidigt samt att användarna redan är bekanta med hjälpmedlen. Detta gör det lättare för användarna att bidra med idéer och delta i utvecklingsarbetet. Löwgren (1993) påstår att många människor anser LoFi-prototypen omfattas av den s.k. 80/20-reglen, vilken innebär att man får 80% vinst på 20% ansträngning, d.v.s. ansträngningen för att ta fram en prototyp är liten jämfört med det betydande resultat man får ut av den.
Ett exempel på en LoFi-prototyp är en s.k. storyboard vars uppgift är visa flödet i ett system, d.v.s. var man hamnar om man klickar på en viss knapp o.s.v.
(Preece, 1994; Löwgren, 1998)

3.1.4 Risker med att utveckla en prototyp

Prototyputveckling är inte bara positiv utan det finns en hel del risker man bör vara medveten om, innan man bestämmer sig för vilken typ av prototyp man ska använda.

- För avancerad prototyp – kunden tror att prototypen är det slutliga systemet och kan inte förstå varför man lägger ner ytterligare 3 månader för att ”göra klart”. Varför ska de betala för något som i deras ögon redan finns?
  De kanske inte förstår att prototypen är ett slags ”skal”, där t.ex. säkerhet, integration och total funktionalitet saknas. (von Dorrien, 2001)

- För enkel prototyp – kunden tror att prototypen kommer att se ut på det enkla sättet och får en negativ inställning till systemet innan det ens har tagits i bruk.
  (von Dorrien, 2001)

- Det finns en risk att utvecklaren resp kunden låser sig för tidigt vid en viss lösning och därför kan det bli svårt att tänka i andra banor. (von Dorrien, 2001)

- En annan risk med prototyper är att ineffektiv prototypkod ibland kan återanvändas vilket kan leda till att det slutliga systemets prestanda sjunker.
  P.g.a. tidsbrist eller skenande kostnader finns risken att en alltför fungerande prototyp ”tillfälligt” tas i bruk som det slutliga systemet. (Sommerville, 2001)

4 Definition av RUP

4.1 Utvecklingen frå n Vattenfallsmodellen till RUP

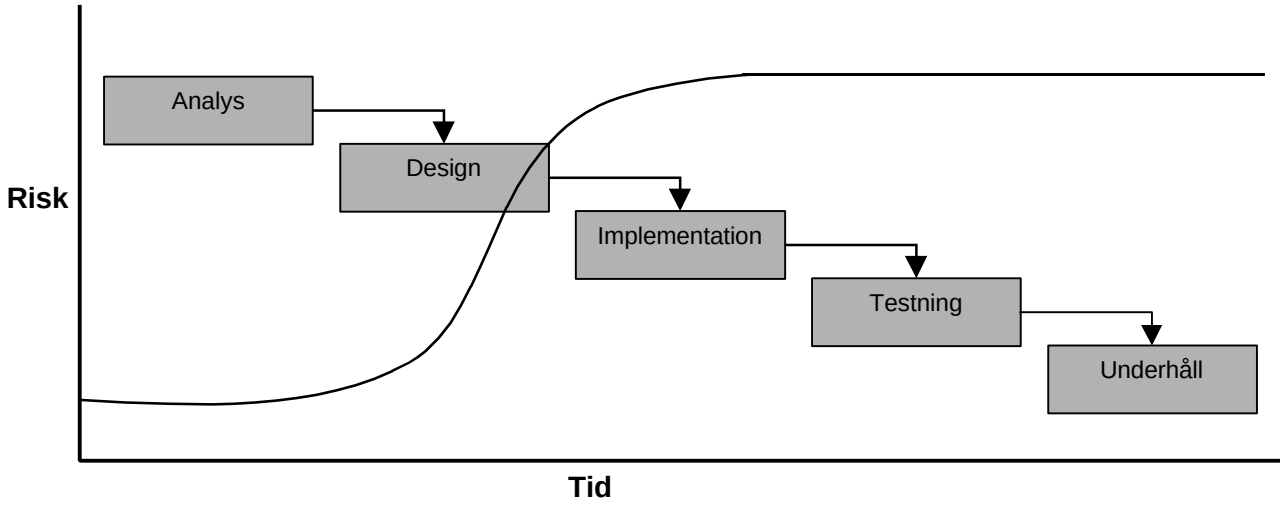
En systemutvecklingsmetod är en samling av generella föreskrifter för hur man ska anpassa datasystem till människors behov. (Mathiassen et al., 1998)

4.1.1 Vattenfallsmodellen

En klassisk systemutvecklingsmetod brukar följa vattenfallsmodellen (se figur 4.1). Modellen tillämpar det som kallas för linjalmetodik. Linjalmetodiken förutsätter att nästa fas inte ska börja innan föregående fas har avslutats. (Kruchten, 2000)

Vattenfallsmodellen innehåller följande fem faser:

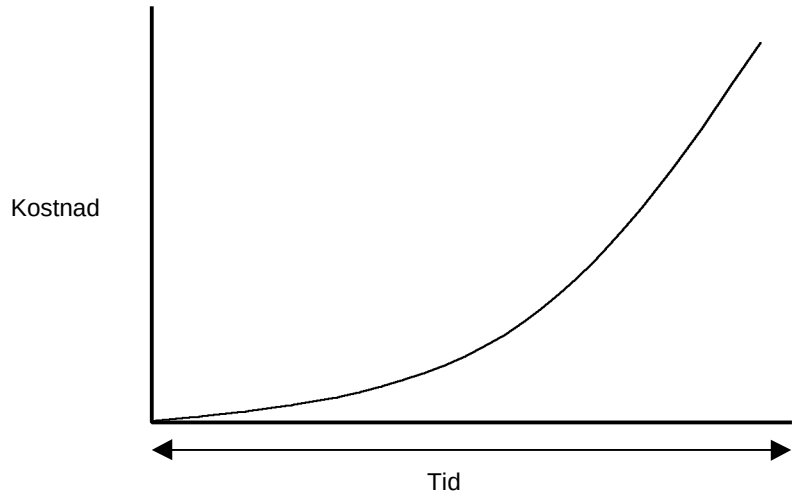
- Analys - definition av kravspecifikation.
- Design - system- och mjukvarudesign.
- Implementation - implementation och enhetstestning.
- Testning - integration och systemtestning.
- Underhåll – drift och underhåll.



Figur 4-1: Vattenfallsmodellen kompletterad med när i tiden risker upptäcks (Sommerville, 1991, s. 45; Kruchten, 2000, s. 6).

Det fundamentala problemet med vattenfallsmodellen är att den har en tendens att gömma de riktiga riskerna i ett projekt tills det är försent att åtgärda dem på ett meningsfullt sätt. Med andra ord är vattenfallsmodellen inte byggd för att förutse och på ett bra sätt ta hand om de risker och misstag som kan förekomma tidigt i processen.

Kostnaderna blir högre och högre ju senare misstag från tidigare faser upptäcks.
(Se figur 4-2).



Figur 4-2: Ju längre tiden går innan man hittar misstag desto högre blir kostnaden för att åtgärda dem. (Kruchten, 2000, s. 13).

Ett annat problem med vattenfallsmodellen är att kravspecifikationen för det nya systemet fryses tidigt i processen. Risken med detta är att kunden får ett system som inte fungerar som han vill ha det.

4.1.2 Spiralmodellen

Som svar på vattenfallsmodellens svårigheter att hantera och kontrollera olika problem i systemutvecklingsprocessen utvecklade Barry Boehm den s.k. ”spiralmodellen”.
(Preece, 1994)

Istället för att som i vattenfallsmodellen utföra aktiviteterna i en sekvens, d.v.s. gå från en aktivitet till en annan när den föregående har avslutats, tog Boehm fram en process som utförs som i en spiral. Varje iteration motsvarar en fas i systemutvecklingsprocessen.

En viktig skillnad mellan spiralmodellen och tidigare systemutvecklingsmetoder är att man i spiralmodellen fokuserar på att hitta risker i projektet. Oidentifierade risker kan leda till stora problem senare i projektet, t.ex. att det drar över tiden och att kostnaderna inte går att hålla inom budgetens ramar. (Sommerville, 2001)

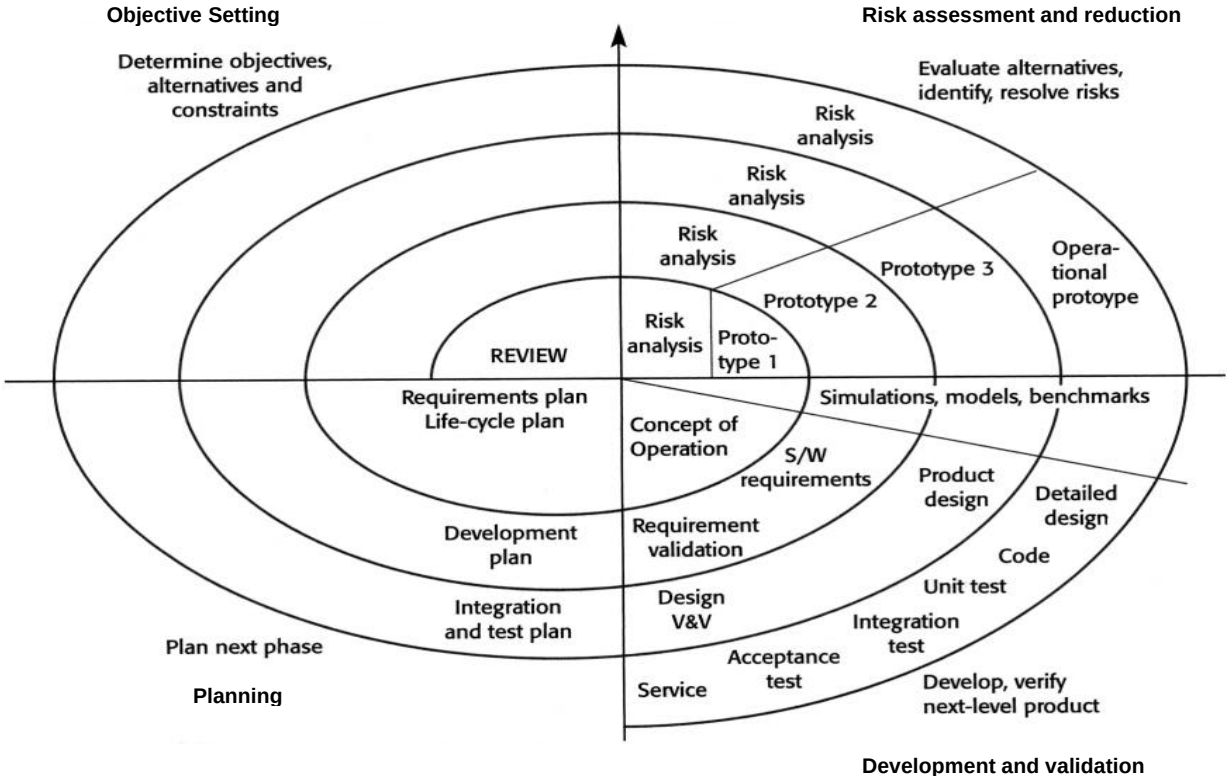
Varje loop i spiralen delas in i fyra sektorer. (Sommerville, 2001)

1. *Fastställande av mål (objective setting)* – i den första sektorn fastställer man specifika mål för resp. fas i projektet. Man fastställer även begränsningar, d.v.s. vad systemet inte ska kunna hantera. En detaljerad projektplan för projektledningen tas fram.
   Risker i projektet identifieras och alternativa strategier, beroende på dessa risker, planeras.

2. *Riskvärdering och riskreduktion (risk assessment and reduction)* – i den andra sektorn genomförs en detaljerad analys av varje risk som har identifierats. Åtgärder vidtas för att reducera riskerna genom att t.ex. utveckla en prototyp.

3. *Utveckling och validering (Development and validation)* – i den tredje sektorn, efter riskutvärderingen, väljer man en utvecklingsmodell för systemet.

4. *Planering (Planning)* – i den fjärde sektorn granskas projektet och beslut fattas om man ska gå vidare till nästa loop i spiralen. Om så är fallet upprättas planer för nästa fas i projektet, annars itererar man ännu ett varv inom samma loop.



Figur 4-1: Boehms spiralmodell av systemutvecklingsprocessen (Sommerville, 2001, s. 54).

Man börjar med att tänka igenom målsättningen för varje cykel i spiralen, t.ex. prestanda, funktionalitet. Möjligheter för att uppnå dessa mål och begränsningar för varje alternativ räknas upp. Därefter bedöms varje alternativ mot de tidigare fastställda målen. Detta brukar resultera i att man identifierar riskkällor tidigt i projektet. Nästa steg är att utvärdera dessa risker genom aktiviteter som t.ex. mer detaljerad analys, prototyputveckling eller simulering. När riskerna slutligen har bedömts, utvecklar man lite till och därefter följer planeringsaktiviteter för nästa fas i processen.
(Sommerville, 1998)

Spiralmodellen använder sig av samma huvudprocesser som vattenfallsmodellen, d.v.s. analys, design, implementation, testning samt underhåll, men med den skillnaden att flera iterationer krävs inom dessa huvudprocesser.

En av de viktigaste nyheterna är att spiralmodellen introducerar idén med prototyputveckling inom varje fas för att bättre kunna förstå kraven inför nästa iteration genom att identifiera de komponenter som har den största risken att vara fel eller som kostar mest att i efterhand rätta till. (Preece, 1994)

4.1.3 Vad är RUP och varför RUP?

Resten av kapitel 4 baseras på Kruchten (2000) samt RUP Online (2001).

Under årens lopp har olika it-projekt misslyckats p.g.a. olika orsaker, bl.a. missförstånd när det gäller användarnas behov, oförmåga att hantera föränderliga krav, sena upptäckter av projektbrister, projektmedlemmar som står i vägen för varandra m.fl. Det har varit svårt att hålla reda på vem som ändrade vad, när och varför.

RUP bygger på spiralmodellens idéer, som vi tidigare har nämnt, bl.a. fokuserar på att hitta och åtgärda risker så tidigt som möjligt i systemutvecklingsprocessen och hantera nytillkomna krav.
RUP har ambitionen att tillhandahålla ett mer disciplinerat sätt att fördela arbetsuppgifter och ansvarsområden inom en utvecklingsorganisation. Målet med RUP är att säkerställa att produktion av mjukvara sker med hög kvalitet som möter slutanvändarnas behov samt att det följer en förutbestämd tidsplan och budget. RUP möjliggör för projektledning att överblicka och kontrollera ett projekt.

Rational har samtidigt utvecklat olika verktyg som ska stödja systemutvecklingsaktiviteter och underlätta utvecklarnas arbete. På detta sätt håller sig RUP inte bara på en teoretisk nivå utan ger även stöd i praktiken.
Vi går inte närmare in på verktygen utan håller oss till den teoretiska delen.

RUP är mycket omfattande och det skulle krävas en hel uppsats i sig för att i detalj beskriva RUP. Vi försöker därför att på en mycket grundläggande nivå förklara RUP:s grundstruktur och uppbyggnad i resterande delar av detta kapitel.

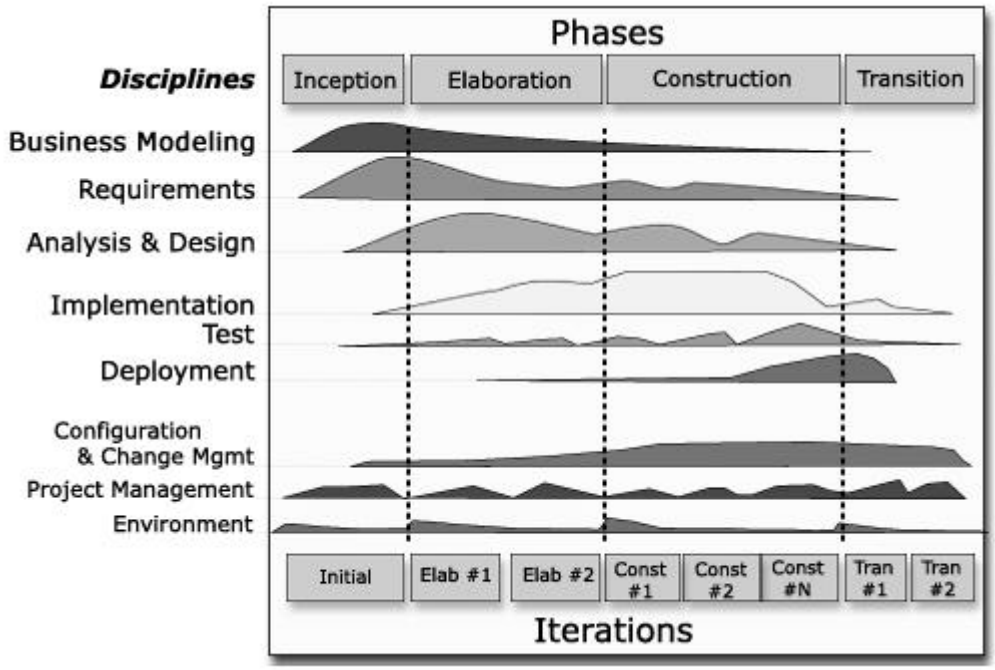
4.2 RUP:s uppbyggnad

RUP har två dimensioner:

- den *horisontella axeln* motsvarar tid och visar livscykelns aspekter på utvecklingsmetoden alltmedan den utvecklas.
- den *vertikala axeln* motsvarar arbetsflödet, som grupperar aktiviteter inom sitt område.

Den första dimensionen motsvarar en dynamisk syn av utvecklingsmetoden. Den uttrycks i termer av cykler (cycles), faser (phases), iterationer (iterations) och milstolpar (milestones).

Den andra dimensionen motsvarar en statisk syn på utvecklingsmetoden. Den i sin tur uttrycks i termer av processkomponenter, aktiviteter, arbetsflöden, artefakter och roller.

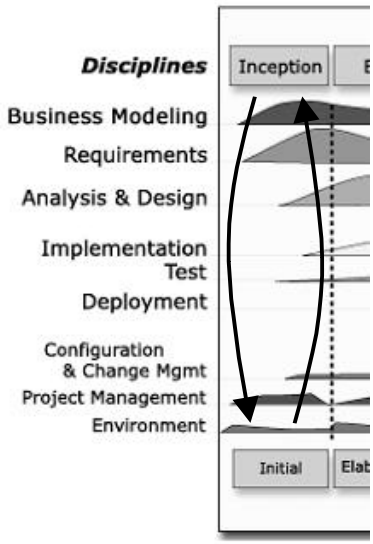


Figur 4-1: RUP:s arbetsflöden, faser och iterationer (Kruchten, 2000, s. 23).

4.2.1 Den horisontella dimensionen

Den horisontella dimensionen är ett medel för projektledningen att styra och följa upp utvecklingen av projektet.
Utvecklingsmetodens livscykel är indelad i fyra sekventiella faser: ”Inception”, ”Elaboration”, ”Construction” och ”Transition”.
I slutet av varje fas finns en avstämningspunkt (milestone) för att fastställa att målen för just den fasen har uppnåtts.[5] Om målen inte motsvarar de krav man uppställt för fasen, itererar man ytterligare en gång inom samma fas (iterations). Först när en tillfredsställande utvärdering har uppnåtts tillåts projektet att gå vidare till nästa fas.

[5] Lifecycle Milestone Review, enligt RUP



Figur 4-1: Iteration inom en fas.

4.2.1.1 Inception-fasen

Mål

Målet med inception-fasen är att projektets intressenter ska få en gemensam syn på projektets problemområde. För nya projekt innebär detta att risker inom problemområdet och systemkrav måste identifieras innan projektet kan fortsätta.
När det gäller förbättringar på redan existerande system, är inception-fasen kortare, men den är fortfarande fokuserad på att säkerställa att projektet är värt att genomföra och att det är genomförbart.
De främsta målen för inception-fasen är:

- att fastställa ett projekts omfattning, d.v.s. vad som skall ingå resp inte ingå i projektet.
- att ta fram de mest kritiska och viktiga användarscenarierna.
- att visa och kanske demonstrera minst en arkitektur baserad på något av användarscenarierna.
- att uppskatta projektets kostnader och ta fram ett planeringsschema för hela projektet.
- att uppskatta potentiella risker.
- att förbereda stöd för projektet.

Aktiviteter

Aktiviteter som ingår i inception-fasen är:

- att formulera projektets omfattning.
- att planera och förbereda administration kring projektet, t.ex. riskhantering och bemanning.
- att implementera ett användarscenario för att säkerställa att projektet är genomförbart.
- att fastställa projektet och dess organisation, välja verktyg, besluta om vilka delar i processen som behöver förbättras.

Resultat
Resultatet av inception-fasen är skapandet av följande artefakter:

- ett dokument som presenterar en generell vision av projektets viktigaste krav och huvudsakliga begränsningar.
- lista över alla användarscenarier och aktörer som kan identifieras så pass tidigt i projektet.
- en initial projektordlista.
- en initial affärsmodell.
- initial riskutvärdering.
- en projektplan som visar faser och iterationer.

4.2.1.2 Elaboration-fasen

Syftet med elaboration-fasen är att analysera problemområdet, etablera en fullgod arkitekturell grund, utveckla projektplanen samt att eliminera högriskområden inom projektet. Elaboration-fasen är den mest kritiska av de fyra faserna eftersom det är här man fattar beslut om projektet ska fortsätta eller om det ska läggas ner.

Mål
De främsta målen för elaboration-fasen är:

- att definiera, validera arkitekturen så snabbt som det är praktiskt möjligt.
- att fastställa visionen.
- att fastställa en högkvalitativ konstruktionsplan för constructions-fasen.
- att demonstrera att den fastställda arkitekturen kommer att stödja visionen till en rimlig kostnad inom rimlig tid.

Aktiviteter
Aktiviteter som ingår i elaboration-fasen är:

- att utarbeta visionen och upprätta en solid förståelse för de mest kritiska användarscenarierna.
- att utarbeta processen, infrastrukturen och utvecklingsmiljön och sätta processen, verktygen och supporten på plats.
- att utarbeta arkitekturen och välja ut komponenter.

Resultat
Resultatet av elaboration-fasen är skapandet av följande artefakter:

- en use-casemodell.
- kompletterade krav.
- arkitekturbeskrivning för mjukvaran.
- en körbar arkitekturell prototyp.
- en reviderad lista med risker.
- en utvecklingsplan.
- en preliminär användarmanual.

4.2.1.3 Construction-fasen

Under construction-fasen utvecklar man alla återstående komponenter och integrerar dem i produkten. Alla funktioner testas noggrant. Fasen kan också ses som en tillverkningsprocess där man fokuserar på att hantera resurser och kontrollera verksamheten för att optimera kostnader, tidsplaner och kvalitet.

Mål

De främsta målen för construction-fasen är:

- att minimera utvecklingskostnaderna genom optimering av resurser och genom att undvika onödiga omarbetningar.
- att uppnå tillfredsställande kvalitet så snabbt som det är praktiskt möjligt.
- att uppnå användbara versioner (alpha, beta och andra testutgåvor) så snabbt som det är praktiskt möjligt.

Aktiviteter

Aktiviteter som ingår i construction-fasen är:

- att hantera resurser och optimera processen.
- att komplett utveckla komponenter och testa mot de definierade evalueringskriterierna.
- att utvärdera produktutgåvor gentemot visionens acceptanskriterier.

Resultat

Construction-fasen resulterar i en produkt färdig att lämnas till slutanvändarna som minst innehåller:

- mjukvara integrerad till de adekvata plattformarna.
- användarmanualer.
- en beskrivning av den aktuella utgåvan.

4.2.1.4 Transition-fasen

Syftet med transition-fasen är att förflytta mjukvaran till användarnas miljö. Efter att produkten har levererats till kunden brukar en del problem uppstå som kräver att man utvecklar nya utgåvor, rättar till vissa problem eller avslutar tidigare uppskjutna funktionella komponenter.

Mål

De främsta målen för transition-fasen är:

- att uppnå att användarna kan ge support till sig själva.
- att uppnå att intressenterna har en gemensam syn på delar av systemet som ska tas i drift och att det överensstämmer med visionens evalueringskriterier.
- att uppnå att ”utrullningen” av det slutliga systemet ska ske så snabbt och så kostnadseffektivt som det är praktiskt möjligt.

Aktiviteter

Aktiviteter som ingår i transition-fasen är:

- Marknadsföring, försäljning samt utbildning av personal på plats.
- Finjusteringsaktiviteter, t.ex. fixa buggar och förbättringar av prestanda och användbarhet.
- Utvärdera de delar av systemet som ska tas i drift mot visionen och acceptanskriterierna för produkten.

4.2.2 Den vertikala dimensionen

RUP delar upp processen i nio discipliner (disciplines). Dessa nio discipliner delas i sin tur in i sex "Engineering workflows" (Business Modeling, Requirements, Analysis & Design, Implementation, Test och Deployment) och i tre stycken "Supporting workflows" (Configuration and Change Management, Project Management, Environment).

En disciplin visar alla aktiviteter som kan genomföras för att ta fram vissa artefakter. På en mer detaljerad nivå -" workflow details"- visas hur roller samarbetar, hur dessa roller använder sig av och tar fram artefakter. Längre fram beskriver vi mer om roller och workflows. (RUP Online, 2001)

4.2.2.1 Business Modeling

Syftet med Business Modeling-disciplinen är att:

- förstå strukturen och dynamiken i en organisation för vilken systemet skall utvecklas.
- förstå de aktuella problemen i organisationen och identifiera potentiella förbättringar.
- säkerställa att kunden, slutanvändarna och utvecklarna har en gemensam uppfattning av organisationen.
- ta fram de systemkrav som behövs för att stödja organisationen.

4.2.2.2 Requirements

Syftet med Requirements-disciplinen är att:

- upprätta och underhålla överenskommelser med kunder och andra intressenter om vad systemet ska kunna utföra.
- tillhandahålla en bättre förståelse av systemkraven för systemutvecklarna.
- definiera systemets begränsningar.
- ta fram ett underlag för planering av iterationernas tekniska innehåll.
- ta fram ett underlag för uppskattning av kostnader och tid för att utveckla ett system.
- definiera ett användargränssnitt för systemet som fokuserar på användarnas behov och mål.

4.2.2.3 Analysis and Design

Syftet med Analysis & Design-disciplinen är att:

- omvandla kraven till en design av det kommande systemet.
- utveckla en stabil systemarkitektur.
- anpassa designen till implementationsmiljön.

4.2.2.4 Implementation

Syftet med Implementation-disciplinen är att:

- fastställa organisationen av koden.
- implementera klasser och objekt.
- testa de utvecklade komponenterna som enheter.
- integrera resultatet av individuella implementatörers eller gruppers arbete (enheterna) till ett exekverbart system.

4.2.2.5 Test

Syftet med test-disciplinen är att:

- verifiera interaktionen mellan objekt.
- verifiera en ordentlig integration av mjukvarans komponenter.
- verifiera att alla krav har implementerats på rätt sätt.
- identifiera och säkerställa att felaktigheter uppmärksammas innan deployment-fasen.

4.2.2.6 Configuration and Change Management

Syftet med Configuration and Change Management-disciplinen är att:

- Hantera och kontrollera kravförändringar och underhålla projektartefakterna.

4.2.2.7 Deployment

Syftet med Deployment-disciplinen är att:

- beskriva de aktiviteter som säkerställer att mjukvaruprodukten finns tillgänglig för slutanvändarna.

4.2.2.8 Project Management

Syftet med Project Management-disciplinen är att:

- hantera risker, komma över de hinder som finns i vägen för att framgångsrikt leverera en produkt som möter kundernas behov.
- tillhandahålla ett ramverk för att hantera mjukvaruintensiva projekt.
- tillhandahålla praktiska riktlinjer för planering, bemanning, utföra och överblicka projekt.
- tillhandahålla ett ramverk för riskhantering.

4.2.2.9 Environment

Syftet med Environment-disciplinen är att:

- tillhandahålla de verktyg som krävs för att stödja hela processen .
- fokusera på aktiviteter som är nödvändiga för att konfigurera processen för ett projekt.

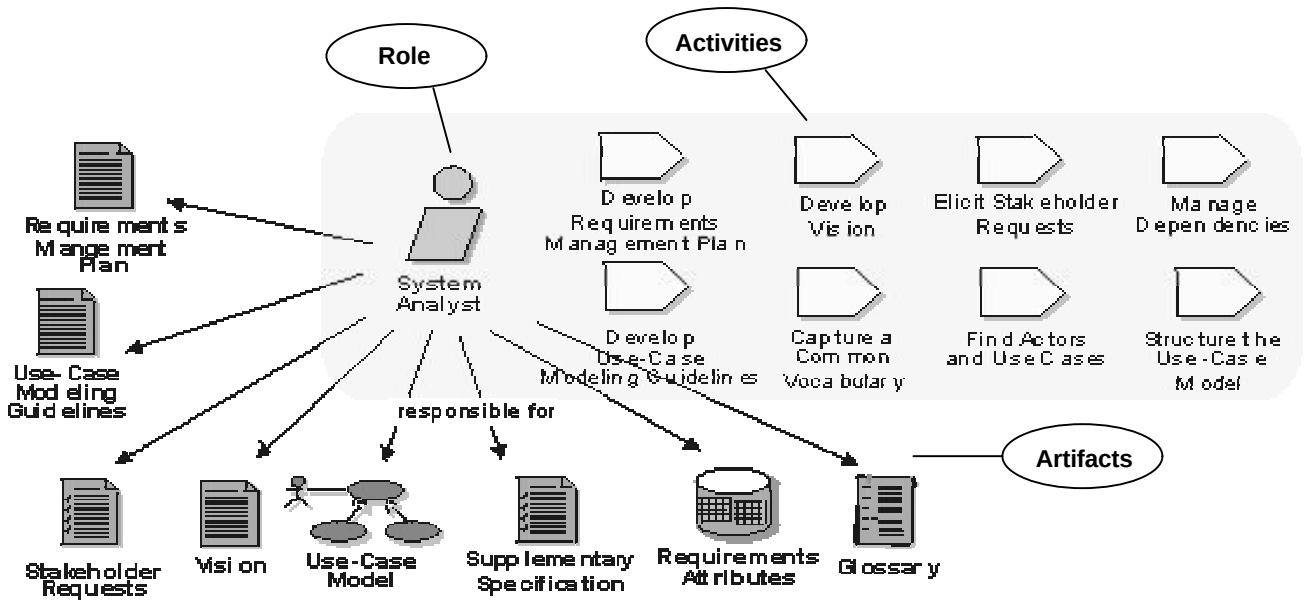
4.3 Hur jobbar man med RUP?

En systemutvecklingsmetod beskriver *vem* som gör *vad*, *hur* och *när* det görs i ett projekt. RUP använder sig av fyra modellelement.

- Roll (role): Vem utför uppgiften?
- Aktivitet (activity): Hur ska uppgiften utföras?
- Artefakt (artifact): Vad ska utföras?
- Arbetsflöde (workflow): När ska det utföras?

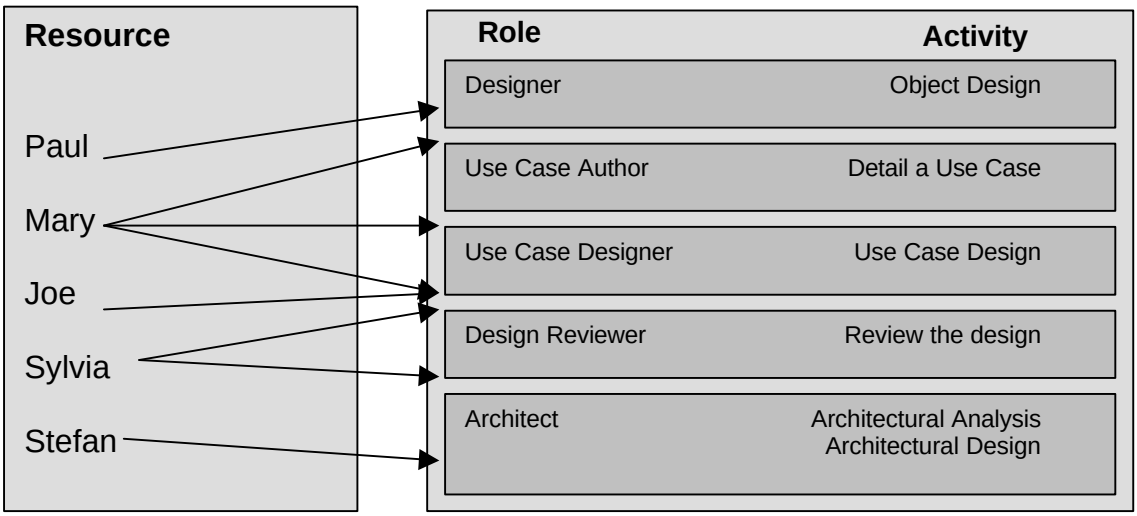
En roll definierar beteende och ansvarsområde för en individ eller för en grupp av individer som arbetar tillsammans i ett lag. Beteendet uttrycks genom de *aktiviteter* rollen ska utföra och ansvarsområdet uttrycks genom de *artefakter* rollen skapar, modifierar och kontrollerar. En artefakt är information som har skapats, modifierats eller använts i en utvecklingsprocess. Det kan t.ex. vara en use-casemodell, ett dokument eller en källkod. Artefakter används som indata för att rollen ska kunna utföra sina aktiviteter och dessa aktiviteter resulterar i artefakter som i sin tur blir indata för nästa roll.

Nedanstående bild visar de olika aktiviteter en ”System Analyst” deltar i och de artefakter som dessa aktiviteter resulterar i.



Figur 4-1: Rollbeskrivning, aktiviteter och artefakter för en System Analyst (RUP Online, 2001).

Inom RUP finns 31 roller definierade, t.ex. system analyst, designer, test designer m.fl. Observera att roller inte är individer. En individ kan ha en eller flera roller och varje roll har till uppgift att utföra aktiviteter bestämda för just den rollen.

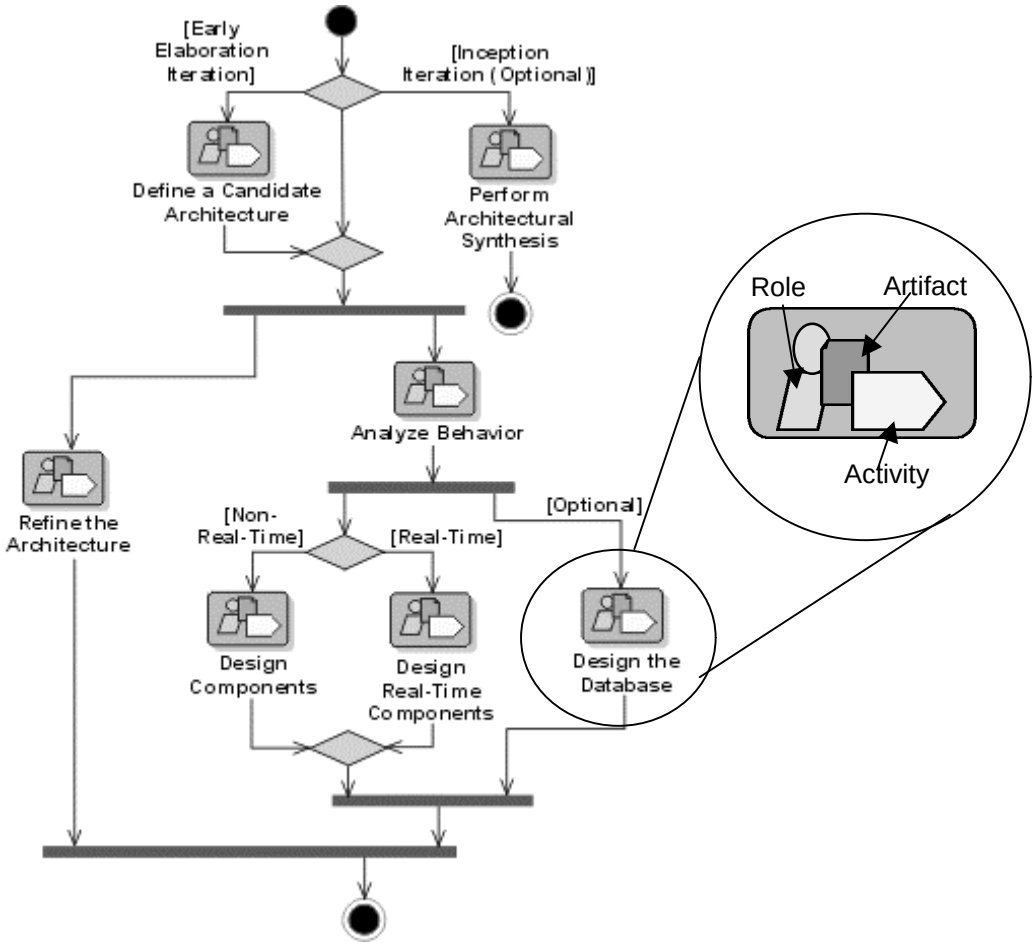


Figur 4-1: En person kan ha flera roller inom RUP (Kruchten, 2000, s. 38).

Det behövs ett sätt att beskriva de aktiviteter som resulterar i värdefull information och som visar interaktion mellan rollerna. Arbetsflöde (workflow) kallas en sådan sekvens av aktiviteter. RUP har valt att organisera arbetsflödena enligt ”Disciplines”, ”Workflow details” och ”Iteration plans”:

- ”Disciplines” innebär de discipliner vi tidigare har nämnt.
- ”Workflow details” är ett uttryck för en specifik grupp av relaterade aktiviteter. ”Workflow details” visar även informationsflödet, d.v.s. de artefakter som är indata resp. utdata från och till aktiviteterna.
- ”Iteration plans” är ett sätt att presentera de aktiviteter som ska ske inom en specifik iteration.

Figuren 4-8 visar arbetsflödet inom ”Analysis & Design”.



Figur 4-2: Arbetsflödet inom ”Analysis & Design” (RUP Online, 2001).

4.4 RUP:s syn på prototyper

Vad gäller RUP:s sätt att se på prototyper är de till för att reducera risker och för att reducera osäkerhet när det gäller:

- produktens affärsmässiga livsduglighet.
- stabilitet och prestanda.
- satsning på projektet eller finansiering av det.
- förståelse av krav.
- produktens användbarhet.

En prototyp kan hjälpa till att utveckla stöd för produkten genom att visa något konkret och exekverbart för användare, kunder och ledning.
Målet med prototyper är att de måste förbli tydliga. Om man inte har för avsikt att utveckla prototypen till den slutliga produkten, ska man plötsligt inte anta att prototypen ska bli den slutliga produkten bara för att den fungerar.

RUP har två sätt att se på prototyper:

- Genom vad de *resulterar* i:
  - den ”utforskande” prototypen (throw-away) slängs när den uppnått sitt syfte.
  - den ”evolutionära” prototypen utvecklas stegvis till att bli det slutliga systemet.

- Genom vad de *utforskar*:
  - ”beteendeprototypen” fokuserar på att utforska systemets specifika beteende.
  - den ”strukturella” prototypen utforskar arkitekturella och teknologiska angelägenheter.

Utforskande prototyper

Den utforskande prototypen utvecklas för att testa projektets tveksamheter, antingen funktionalitet eller teknologi eller båda delarna. Den är tillfällig, den görs med minimal ansträngning och syftar till att slängas efter genomförda tester.

Evolutionära prototyper

Den evolutionära prototypen utvecklas från en iteration till nästa. Dess kod omarbetas allteftersom produkten utvecklas. För att kunna hantera omarbetningarna utförs designen och testningarna mer formellt.

Beteendeprototypen

Beteendeprototypen tenderar att vara en utforskande prototyp som fokuserar på vad systemet skall göra sett ur *användarnas* synvinkel (systemets skal).

Strukturella prototyper

Strukturella prototyper tenderar att vara en evolutionär prototyp som fokuserar på vad systemet skall göra ur *utvecklarnas* synvinkel (systemets skelett, arkitektur).

RUP förespråkar prototyputveckling i samband med utvecklingen av användargränssnitt (User Interface Prototype) samt arkitekturell prototyputveckling.

Enligt RUP ska prototyper för användargränssnitt göras i Requirement-disciplinen, både i inception- och elaboration-fasen, med syftet att hantera varierande användbarhetskrav. Prototypen för det användargränssnittet baseras på use case-modellerna, vilka har tagits fram tillsammans med kunden i ett tidigare skede. Ovan nämnda beteendeprototyp är ett lämpligt val vid utvecklingen av en prototyp för användargränssnitt.

När det gäller arkitekturell prototyputveckling görs denna i Analys & Designdisciplinen, både i inception- och elaboration-fasen. Syftet med den arkitekturella prototypen är att implementera de viktigaste arkitekturella besluten som fattats i designfasen. Den arkitekturella prototypen utvärderar arkitekturen och tjänar som en grund för resten av systemutvecklingen.
Denna prototyp är s.k. strukturell och den resulterar i att prototypen, via construction-fasen, slutligen utvecklas till färdiga systemet.

5 Intervjuresultat

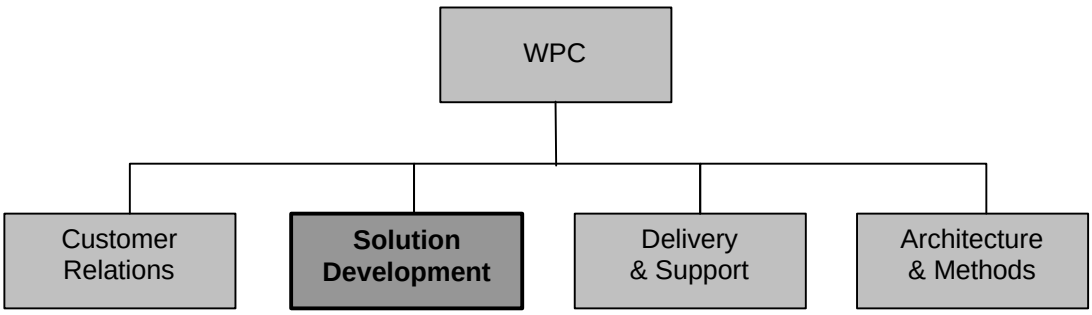
5.1 Introduktion till intervjuer

Vi nämnde i introduktionskapitlet att vi skriver vår uppsats på uppdrag av WPC (Web Program Center) på Volvo IT. Vi har haft fördelen att även ha vår arbetsplats hos WPC under hela vårt uppsatsarbete, vilket har varit mycket fördelaktigt för oss. Främst för att vi lättare har kunnat följa deras utvecklingsarbete, bl.a. genom att delta i deras interna gruppmöten, men även för att vi lättare kunde boka intervjuer och att vi hade de intervjuade nära till hands för extra frågor. Projektet vi har följt är ett utvecklingsprojekt för AB Volvos nya webplats.
Våra litteraturstudier tillsammans med observationer av det fortskridande arbete inom projektet lade grunden till vår undersökning.

Våra intervjuer har inriktats mot tre grupper av projektintressenter, som vi har valt att kalla dem. Vi har försökt att få en objektiv syn på vårt problemområde genom att angripa det från tre olika håll.

1. Vi har intervjuat Runo Burman från avdelningen Methods and Techniques inom Volvo IT. Den avdelningen har bl.a. till uppgift att ge stöd till WPC och andra avdelningar inom Volvo IT beträffande RUP och dess metod. Burman kan betraktas som en ”guru” inom området. Vi är intresserade av metodavdelningens syn på hur RUP bör tillämpas i ett systemutvecklingsprojekt samt tänkandet kring prototyper inom RUP.
   För enkelhetens skull kallar vi fortsättningsvis avdelningen Methods and Techniques för metodavdelningen.

2. Vi har intervjuat utvecklarna på avdelningen ”Solution Development” på WPC, vars uppgift är att utveckla och leverera ett komplett system enligt RUP. Projektgruppen har utformats enligt RUP:s definition av roller och består således av system analytiker, designer, utvecklare av användargränssnitt, systemarkitekter, implementatörer, testare, projektledning samt övriga intressenter (stakeholders). När det gäller WPC har vi intresserat oss för deras sätt att använda/utveckla prototyper samt hur RUP påverkar deras arbete och utvecklingen/användning av prototyperna. Sammanlagt har vi intervjuat tolv personer från WPC.
   Vi vill tydliggöra att vi avser avdelningen ”Solution Development” inom WPC när vi skriver WPC.



Figur 5-1: Organisationsschema över WPC.

3. Vi har intervjuat Charlie Nordblom från AB Volvo som är beställare och kravställare av systemet, således är han kund till WPC. När det gäller honom är vi intresserade av *om* han har en åsikt om RUP och om hans syn på prototyper.

Inför intervjuerna med WPC utformade vi ett antal frågor, närmare bestämt 17 till antalet (se bilaga 1). Tanken med frågorna var inte att ta fram individens personliga erfarenheter av prototyputveckling och RUP utan snarare betona den intervjuade personens uppfattning om sin roll och dess omfattning i projektet. Samtidigt ville vi få fram en helhetsbild av WPC som utvecklingsgrupp.

I intervjuerna med Charlie Nordblom och Runo Burman lät vi dem tala fritt med viss styrning av enstaka frågor från vår sida för att intervjun skulle hålla sig inom vårt intresseområde.

Några av frågorna kanske inte direkt berörde vårt problemområde, men vi valde att ta med dem dels för att leda in intervjupersonen i ämnet, dels för att frågorna även skulle ge oss en helhetsbild. Vissa av frågorna har dessutom varit intressanta som grund för våra framtida förslag kring prototyputveckling tillsammans med RUP.
Intervjuerna genomfördes enskilt för att personerna inte skulle påverkas av varandra. De spelades in och skrevs sedan rent ordagrant för att vi i efterhand lättare skulle kunna studera svaren och se eventuella kopplingar till andra svarsresultat. Det var en ganska slitsam process eftersom själva intervjuerna tog mellan ½timme till 1 ½timme och trots våra flinka fingrar tog det sedan minst två timmar att skriva ut dem.
Dock var det mödan värt, för intervjuerna resulterade i ett omfattande undersökningsmaterial och det var mycket berikande för vår egen förkovran.
De tolv intervjuerna med utvecklarna på WPC har sammantställts i en tabell (se bilaga 2).

5.2 De intervjuades syn på prototyputveckling

Ett av huvudmålen vi hade med intervjuerna var att få fram intressenternas syn på prototyper och prototyputveckling.
Alla de intervjuade har arbetat med prototyper i någon form och de har utvecklat egna uppfattningar om dem. Det första vi ville ha reda på var deras uppfattning om syftet med prototyper samt prototypernas för- och nackdelar.

Runo Burman
Runo Burman anser att många har en dogmatisk syn på prototyper, d.v.s. att prototyper måste innehålla körbar kod och att den ska fungera som en provisorisk version av det slutliga systemet. Själv tycker han inte att det är självklart utan menar att ibland kan en prototyp även vara en snabbskiss som man sedan kastar.
På sin egen fråga *"Ska vi ha en prototyp?"* svarar han: *"Ja, om man behöver det. Är kunderna kända och är deras behov kända är prototyputveckling inte så viktigt*". Det främsta syftet med prototyper, enligt Burman, är att man gör en prototyp för att interagera med kunderna och för att få igång tankarna hos dem.
Under intervjun med Burman framkom inga direkta nackdelar med prototyper.

WPC

På WPC har de olika avdelningsmedlemmarna en blandad uppfattning om vad prototyper och prototyputveckling innebär. Nedan visas de huvudsakliga syftena tillika fördelarna med prototyper, som kom fram under intervjuerna:

- visualisera och visa upp för någon som inte har varit med i diskussionen tidigare.
- ett sätt för kunden att lättare förstå vissa problemställningar.
- ett diskussionsunderlag med kunden.
- en prototyp är något som inte ger sken av att vara den slutliga lösningen. Det ska gå fort, ge snabbt resultat för att stämmas av mot kund. Det kan vara en pappersgrej eller en klickbar variant.
- textdokument svarar på frågan VAD ska göras. Bilder och prototyper svarar på frågan HUR ska det göras.
- testa funktionalitet och idéer. Komma fram till hur man ska utveckla på ett smart sätt.
- prototyper skapar engagemang och ägandeskap hos kunden.
- upptäcka problem på ett tidigt stadium i systemutvecklingsprocessen.

WPC:s uppfattning om nackdelar med prototyputveckling är att:

- kunden kan uppfatta prototypen som det slutliga systemet.
- p.g.a. tidsbrist finns det risk att prototypen blir det slutliga systemet.
- kunderna är ovilliga att betala för prototyper.
- man överarbetar prototypen, d.v.s. man ”pillar med knappar” mest för att det är kul.
- man låser sig för tidigt vid en viss lösning.
- kunden fokuserar på fel saker vid avstämning, t.ex. tittar på färger och ordval istället för funktionalitet.
- man kan lova för mycket med prototyper.
- prototyper är kostsamma.
- de tar tid att göra.

Charlie Nordblom

Charlie Nordbloms åsikt är att prototyper har tre huvudsakliga syften:

1. Att ge honom ett verktyg att kommunicera med sina slutkunder. Där anser han att prototypen är hygglig, men att den inte är bättre än en serie PowerPoint-bilder eller en flash-presentation.

2. Det andra syftet, enligt Nordblom, är att kommunicera mellan leverantör och beställaren. ”*Är detta vad du har tänkt dig eller rättare sagt: så här har vi förstått dina krav och så här är vårt förslag till lösningar, och så visar vi hur det ser ut för dig.*” Där föredrar han storyboards framför prototyper.

3. Det tredje syftet, som Nordblom tror är huvudsyftet för leverantören, är att bygga en prototyp för att testa saker, testa sina resonemang. Han anser att den mest är till för att lösa leverantörens egna problem.
   Denna prototyp, menar Nordblom, är *livsfarlig*” för han är rädd för att ”*få in en massa skräpkod och misslyckade lösningar*” i den slutliga produkten.

Fördelar med prototyper, anser Nordblom, är att man inte ”*bränner så mycket programmeringstid*”, men framför allt ser han det som ett kommunikationsredskap mellan utvecklare och kund. Särskilt om kunden eller beställaren är en ovan beställare som inte förstår funktionaliteten och inte kan läsa en specifikation. Då är prototyp riktigt bra, menar han.
Kan däremot kunden läsa en specifikation passar en prototyp av typen storyboard bättre, tycker Nordblom. Storyboarden är lätt att förstå, lätt att ta till sig och är relativt billig. ”*En storyboard räcker för mig*”, menar han, ”*eftersom den visar det jag är intresserad av, nämligen hur flödet fungerar i systemet. För detta behöver jag inte se en avancerad prototyp.*”

Nordblom brukar ta in en ”challenger” – en tredjepart, som granskar vad det är som gjorts och hur det har gjorts gentemot kravspecifikationen.
En risk med prototyputveckling, menar han, är att dokumentationen kring prototyperna inte håller. ”*Kan man göra en prototyp som är så väldokumenterad att du faktiskt kan sätta en tredjepart på att testa den?*”

Separata frågor till WPC
Till WPC ställde vi frågan om de skulle klara sig utan prototyputveckling i sitt arbete. Svaret var ganska enhälligt. De flesta svarade att det nog skulle gå att arbeta på det sättet, men det är inget man skulle föredra. Många menar också, att hade man inte gjort en prototyp i projektet skulle man ändå göra en för sin egen skull. Två svarade att de klarade sig helt utan.

På frågan om de på WPC har ett behov av en standard när det gäller begreppet prototyp och prototyputveckling svarade de flesta att något slags ”standard” vore önskvärd. Många har upplevt förvirring genom att samma ord nämns, men att man ändå pratar om olika saker. Några menade att en standard kanske inte direkt är nödvändig, men efterlyser ändå något slags bättre definition eller t.o.m. upplysning och utbildning kring begreppet. En ansåg att ur kundperspektivet vore det bra att veta vilka prototyper man kommer att få och vilket format de är i, eftersom alla förstår inte skillnaden mellan en html-prototyp och en klickbar PowerPoint-bild.

5.3 De intervjuades syn på RUP

Ett annat huvudmål med intervjuerna var att få fram intressenternas syn på RUP.
RUP är ett relativt nytt begrepp inom Volvo IT. RUP infördes 1998, men tanken är att det inte ska vara ett ”Big Bang” utan det ska implementeras projekt för projekt. Målet är, enligt Runo Burman, att nå 600 utvecklare på hela Volvo IT under 2002.

Runo Burman
Enligt Burman vill RUP vara en form av standardisering på vad och hur man ska dokumentera. Precis som man har enats om en standard för att beskriva motstånd och andra elektriska komponenter, vill Rational att RUP ska vara standard för systemutveckling.

De två främsta anledningarna till att man dokumenterar inom RUP är, enligt Burman, ”*för att man själv ska komma ihåg*” och ”*för att man vill berätta för någon*”.

Då eliminera risker, planera saker, hantera risker och hantera förändring är huvudmålen med RUP är följaktligen dessa dokument några av de viktigaste att ta fram:

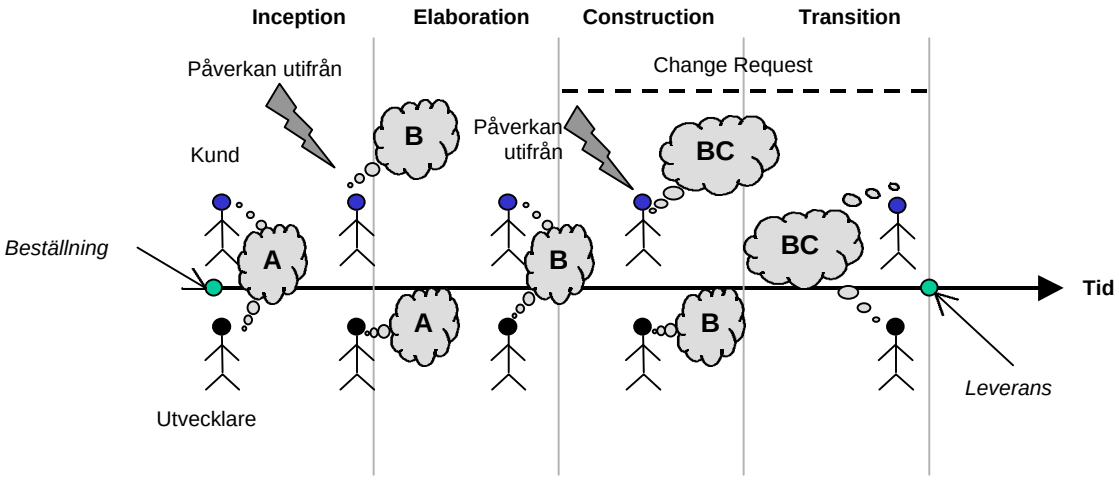
- visionen – en idé om varför man påbörjar projektet.
- risklistan – man för upp de risker man hittar.
- ändringslistan – man för upp de ändringar som tillkommer efter elaboration-fasen (Change Request).

Burman menar att det inte finns någon som använder RUP till 100%.
Man använder så mycket av RUP som man orkar med eller som han själv säger:
”*En elefant äter man ju inte hel, man äter den skiva för skiva*”.

Den största nackdelen med RUP, som Burman ser det, är ”*om man följer den som fan följer bibeln*”. Han menar att de farligaste är de dogmatiska människor som följer skrifterna till punkt och pricka. De minst farliga människorna är de som följer skrifterna lite grand, ”de som tycker att det är trevligt att det finns några regler, men som inte tittar på bokstav för bokstav”.
Den största risken, enligt Burman, är om man äter hela på en gång. ”*Då kvävs man ju direkt*”. Han påpekade återigen varför man gör dokumentationen:
”*Två orsaker: Minnas och tala om för någon annan*”.

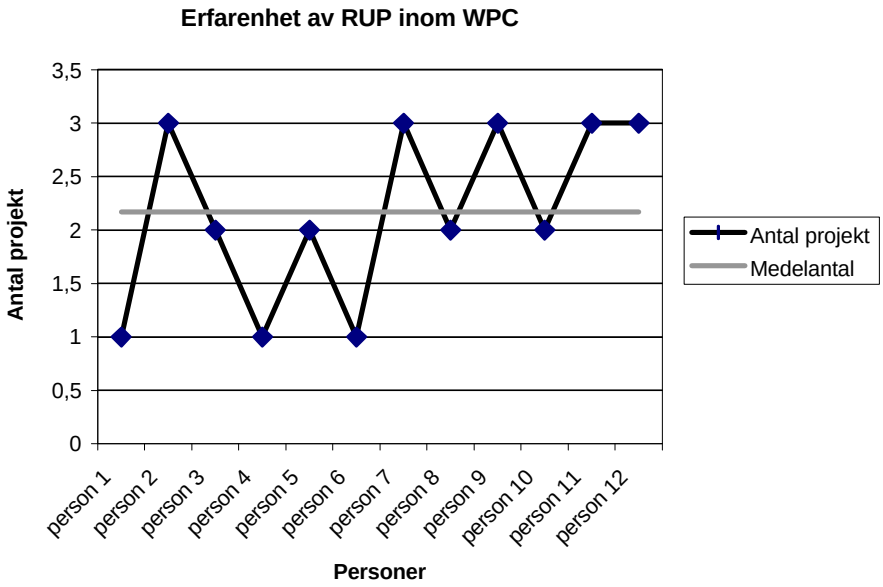
Burman betonar även vikten av kommunikation med kund. Han menar:
”När kunden och jag pratar med varandra så har vi samma idé när vi skiljs åt. Kunden går vidare i tiden och blir utsatt för påverkan utifrån. När vi träffas igen så har vi inte samma bild som han överförde till mig första gången. Han har en helt ny bild. Därför är det viktigt att man interagerar med kunden. Vi måste på något sätt synka våra bilder.
Det vi vill leverera är ju det kunden vill ha i slutet av tidsaxeln, inte det vi bestämde i början. Om man levererar det han vill ha i slutet blir han jättenöjd. Levererar vi det han bestämde tidigare blir han alltid missnöjd. Det är därför det är viktigt att ta ett steg i taget, att verifiera tillbaka ”har idéerna ändrats hos kund?” Det är därför vi ska ha en ”Change Request”- lista, där vi tar med nya krav som vi vill ha in i systemet.”

För att ytterligare betona betydelsen av kommunikation och för att förtydliga för oss under intervjun ritade han nedanstående figur 5-2.



Figur 5-1: Betydelsen av avstämningsmöten mellan kund och leverantör, enligt Runo Burman.

WPC
Då RUP är så pass omfattande som har beskrivits tidigare har Volvo IT gjort en anpassning av RUP så att den ska passa den egna verksamheten. På WPC har man i sin tur gjort en anpassning av Volvo IT-varianten för att passa den egna avdelningens projekt. De flesta på avdelningen har fått genomgå minst en kurs i RUP. M.a.o. vet de vad RUP innebär i teorin, men den praktiska erfarenheten varierar från person till person som nedanstående figur visar. (Se figur 5-3)



Figur 5-2: Erfarenhet av RUP inom WPC.

Trots att den praktiska erfarenheten av RUP på WPC inte är så stor, har avdelningsmedlemmarna många och varierande åsikter om RUP:s för och nackdelar.
De flesta lyfter fram följande som RUP:s främsta styrka:

- Det är ett strukturerat sätt att arbeta på, d.v.s. det förklarar vad man skall göra under hela projektet och hur man kommer att göra det.
- Det är ett bra styrmedel för att få projekten i hamn.
- Det är ett strukturerat sätt att ta fram kommunikationsmodeller mellan kunden och utvecklarna.

Andra viktiga synpunkter som kom fram var att RUP får kunden och utvecklarna att själva tänka efter lite före, d.v.s. redan i den inledande fasen.
RUP har även tydliga mål och tydlig arbetsuppdelning och det är ett bra sätt att dokumentera det man tänker och pratar om.
En avdelningsmedlem framhöll det som positivt att RUP har skapat en egen terminologi fri från värderingar från tidigare systemutvecklingsmetoder.

Många av de intervjuade på WPC uppfattar inte RUP bara som en dans på rosor, utan de har även påpekat RUP:s svagheter.
Det som främst nämns är att RUP är omfattande, komplext och svårt, vilket gör att det tar lång tid att lära sig att arbeta enligt RUP. ”*Man måste ta RUP med måtta för att den inte ska bli alltför betungande*”, menar utvecklarna.

Charlie Nordblom
Kunden, Charlie Nordblom, menar att han inte ser så mycket av själva RUP. Det enda som han framhåller är att man går för snabbt från *vad* till *hur*. Han tycker att det saknas tid för utvärdering och eftertanke i RUP. Det han efterlyser är ett mycket tydligare utvärderande moment innan man börjar göra use-casen och innan man får börja göra prototyper.

5.4 De intervjuades syn på prototyputveckling i samband med RUP

Vårt slutliga mål med intervjuerna var att ta reda på de tre projektintressenternas uppfattning om prototyputveckling enligt RUP.

Runo Burman
Enligt Runo Burman försöker man i elaboration-fasen bevisa att saker och ting går att genomföra. Det gör man oftast i form av en användargränssnittssprototyp, som kanske inte är så avancerad, men den är till för att kontrollera att kunden kan mata in något och få något tillbaka, menar han.

När det gäller arkitekturella prototyper, påstår Burman, att det handlar om hur man ska bygga upp systemet rent tekniskt, d.v.s. med de komponenter som utgör själva systemet. Burman anser att man kan göra en skiss av arkitekturen ganska tidigt (i inception-fasen, vår kommentar), redan när man känner till kraven. ”*Den sitter inte som en smäck direkt*”, säger han, ”*den arkitekturella prototypen visar på hur det kan se ut. Min åsikt är att man inte utvecklar arkitekturella prototyper tillräckligt ofta*”.

Enligt Runo utvecklar man inte någon mer prototyp sedan man har kommit till construction-fasen. Visst det kan finnas något scenario kvar, menar han, men det ska inte vara något av hög risk. Det får inte vara något som påverkar systemets uppbyggnad.

WPC
Punkterna som följer är WPC:s åsikter om RUP på något sätt påverkar prototyputvecklingen och i så fall hur:

- Prototyputveckling utan RUP går fortare än prototyputveckling med RUP, om man inte är väldigt duktig på RUP.
- RUP påverkar prototyputveckling mest negativt eftersom så många inte kan. Man måste jobba med RUP för att bli bättre på det.
- RUP fungerar bra om man har de rätta förutsättningarna: gott om tid, resurser och ekonomi.
- Ju större projekt, desto större nytta .
- RUP säger att man ska jobba med prototyper främst i elaboration-fasen eftersom det är där man testar och experimenterar.
- Positivt att det i RUP byggs in det här med körbar arkitektur och releaser efter varje iteration. Före RUP var det inte så releaseorienterat. Nu vet man i varje iteration vad som ska göras och vad nästa release ska innehålla.
- I och med att RUP förespråkar iterationer blir det automatiskt så att man gör någon form av prototyp.

- Ser ingen direkt koppling.
- Det är möjligt att det hade sett annorlunda ut utan RUP. Jag kan inte avgöra det.

Charlie Nordblom
Under intervjun med Charlie Nordblom framkom inga synpunkter i denna fråga.

6 Diskussion och Slutsatser

6.1 Prototyputveckling

Av våra litteraturstudier framkommer tydligt att prototyper är ett hjälpmedel vid systemutveckling. Prototyper kan, som bekant, användas för riskeliminering, testning, kommunikation, samt som beslutsunderlag och utbildningsunderlag.

En prototyp behöver inte vara en provisorisk version av en produkt, som t.ex. Anders Lotsson påstår (2001). Sådana och liknande ”hårda” uttalanden bidrar till den allmänna förvirring som idag råder kring begreppet prototyp.
Våra empiriska studier visar att just en sådan förvirring existerar. Ingen av de intervjuade personerna i vår undersökning hade exakt samma uppfattning om vad prototyper och prototyputveckling egentligen innebär.

Vad gäller ”när” man ska använda prototyper är bilden fortfarande väldigt suddig. De flesta författarna i vår litteraturstudie uttalar sig ungefär som Runo Burman gör: ”*Man använder prototyper när man behöver dem*”. Det kan ju låta förnuftigt och ganska självklart, men vem fattar beslutet om att utveckling av prototyper ska ske?

Den rådande begreppsförvirring gör att man på beslutsfattande nivå inte planerar in utveckling av alla typer av prototyper. Undantaget är användargränssnittsprototyper.
Vår studie visar dock att hos utvecklarna finns behovet att utveckla prototyper. Många av dem gör prototyper vid sidan om, eftersom behovet finns, för att testa sina idéer om hur de ska konstruera eller designa en viss del av systemet. Beslutet för sådana prototyper fattas snabbt av varje enskild utvecklare och oftast valideras prototypen av sin egen skapare. Faran med detta är att utvecklaren kan låsa in sig själv i sin egen lösning, vilket innebär att han inte är den mest optimale att utvärdera prototypen.
Beträffande användargränssnittsprototyper beslutas det om dem redan tidigt i projektet. Resurser för genomförandet av prototypen planeras medvetet in till skillnad från de ”egenbeslutade” prototyperna. När det gäller användargränssnittsprototypen kan man bestämma vilken utvecklingsmetod och teknik som ska användas till prototypen samt planera in olika ”deadlines” för utvärdering av prototypen.

En annan konsekvens av begreppsförvirringen är att kunden kan få en negativ inställning till prototyper. I vårt fall säger kunden att han inte vill ha en funktionell prototyp, utan han nöjer sig med en storyboard. Detta påstående bekräftar än en gång förvirringen kring begreppet prototyp. En storyboard är inget mindre än en prototyp, men en LoFi sådan. Vår uppfattning är dock att kunden menade att han inte vill ha en kostsam HiFi-prototyp bara för att visa flödet i systemet, utan att det i detta fall räckte med en storyboard.

Vi identifierade två olika perspektiv när det gäller för- och nackdelar med prototyputveckling: systemutvecklarnas resp. kundens perspektiv. Det framgår av båda perspektiven att fördelen med prototyputveckling är att prototyper tjänar bäst som diskussionsunderlag mellan kund och leverantör för att båda ska ha en gemensam bild av det framtida systemet.

När det gäller syftet att testa funktionalitet och olika lösningar har de dock skilda uppfattningar.
Systemutvecklarna menar att det är en fördel för dem att testa funktionaliteten samt att identifiera risker i systemet. Från kundens sida ses dock detta som en nackdel, då dessa tester m.m., enligt kunden, endast är till för att lösa utvecklarnas egna problem. Han menar att man riskerar att ”få in en massa skräpkod och misslyckade lösningar” i den slutliga produkten.
När det gäller nackdelar med prototyputveckling ur systemutvecklarperspektiv kan kunden uppfatta prototypen som det slutliga systemet, att man som systemutvecklare kan låsa sig för tidigt vid en viss lösning samt att kunden kan fokusera på fel saker vid avstämning.

Sammanfattningsvis visar vår undersökning att den allmänna uppfattningen om prototyper inom WPC, avdelningen Methods and Techniques och hos kunden är att förvirring kring begreppet prototyp råder. På WPC uttrycks desstuom att ett behov av att tillämpa prototyper finns samt att en gemensam definition av begreppet prototyp vore önskvärd.

Beträffande vilka tydliga fördelar som finns när det gäller prototyputveckling uttrycker både kunden och systemutvecklarna att prototyper är ett bra verktyg för kommunikation/avstämning mellan kund och leverantör.
En ytterligare fördel som framkommer för systemutvecklarna är att prototyper kan användas för testning och riskeliminering. Denna fördel ses dock som en nackdel ur kundens perspektiv.

Ur systemutvecklarnas perspektiv ses nedanstående som nackdelar med prototyputveckling:

- prototypen kan tolkas som det slutliga systemet av kunden.
- risk att systemutvecklare låser sig för tidigt vid en viss lösning.
- risk att kunden att fokuserar på fel saker vid avstämning.

6.2 RUP

Som vi tidigare har nämnt är RUP en systemutvecklingsmetod som de senaste åren har börjat sprida sig i systemutvecklingskretsar.

Vår tanke när vi formulerade intervjufrågorna för den empiriska studien, var att intervjupersonerna på WPC skulle svara utifrån sin RUP-definierade roll och inte som enskild person. Resultat blev inte riktigt som vi hade tänkt oss av två anledningar:

- Att RUP inte helt har slagit igenom som systemutvecklingsmetod på WPC.
- Att våra frågor inte betonade deras RUP-roll tydligt nog.

Av vår empiriska studie framgår att systemutvecklarna på WPC har deltagit i genomsnitt två projekt, där RUP har använts som systemutvecklingsmetod. De flesta av systemutvecklarna har dessutom genomgått en RUP-kurs.
Med dessa fakta i åtanke vill vi påvisa att erfarenhet i det här fallet inte har någon större betydelse när det gäller ”acceptansen” av RUP. Studien visar snarare att de systemutvecklare som har mindre vana av systemutvecklingsarbete har tagit till sig RUP på ett lättare sätt än de mer erfarna systemutvecklarna. Vi tror att det beror på att de mer erfarna systemutvecklarna har utvecklat ett eget effektivt sätt att arbeta.

En övergång från det gamla sättet att arbeta till det nya kan vara ineffektivt och påfrestande för utvecklarna.

Det som framkom under intervjuerna som den största nackdelen med RUP var att det upplevdes som alltför tungt och krävande p.g.a. all omfattande dokumentation.
Vi menar att det återigen har att göra med att personlig erfarenhet av RUP saknas, men även att det inte har planerats in extra tid för dokumentation i projektet.
Istället körs två parallellspår: ett där man utvecklar som man tidigare har gjort och ett där man försöker uppfylla RUP:s krav på dokumentation.

RUP är också uppbyggd för att ge stöd både till systemutvecklare och till projektledning, vilket tidigare systemutvecklingsmetoder inte har betonat lika tydligt, om de överhuvudtaget har betonat det. Det handlar främst om stöd i form av en tydlig rollbeskrivning av arbetsområdet för systemutvecklarna och stöd i form av avstämningspunkter mellan faserna, riskhantering m.m. för projektledning.

En möjlig fälla med RUP:s rollindelningar är att de kan innebära försämrad kvalitet i det pågående projektet. Som vi tidigare har nämnt är utvecklarna koncentrerade på sina arbetsuppgifter och producerar egna artefakter, vilka blir input till andra utvecklare när de i sin tur skapar sina artefakter. Detta innebär att alla utvecklare är mer eller mindre beroende av varandra. En utvecklare kan alltså komma att utlösa en seriereaktion genom att producera artefakter av sämre kvalitet.

RUP:s ambition är att utveckla ett bra samarbete mellan utvecklare, ledning och kund. Det försöker man uppnå genom att utvecklarna och ledning inte bara ska *berätta* om vad man har gjort. Med dokumentation ska de också *bevisa* vad de har gjort.
RUP kräver mycket mer dokumentation än vad någon annan systemutvecklingsmetod har gjort hittills. Vi befinner oss i ett samhälle som ständigt och allt snabbare förändras, vilket även innebär att system utvecklas mer än någonsin tidigare. En viktig faktor för framtida underhåll och vidareutveckling av dessa system är just att man dokumenterar flitigt.

Sammanfattningsvis visar vår undersökning att den allmänna uppfattningen inom WPC och på metodavdelningen är att RUP är tungt, men det gäller att välja ut de bitar som passar för just det specifika projektet, vilket både Kruchten och Burman betonar. Anledningen till att RUP är så omfattande är att det ska kunna passa till så många typer av projekt och därför ges inte alltid svaren på hur man ska göra vid ett visst problem. RUP är ett stöd, men det måste anpassas till olika typer av projekt och man måste ha en viss erfarenhet av RUP för att kunna tillgodogöra sig RUP maximalt.
Kundens allmänna uppfattning är att det går för fort från *vad* man ska göra till *hur* man ska göra det. Han menar att det behövs mycket tydligare utvärderande moment innan man börjar göra use-casen och innan man får börja utveckla prototyper.

6.3 Prototyputveckling i samband med RUP

Vår första uppfattning om prototyputveckling inom RUP var att den inte var heltäckande när det gäller prototyputveckling. Vi tyckte att det saknades tydliga riktlinjer för hur man skulle tillämpa prototyper inom denna systemutvecklingsmetod. Efter litteraturstudier och intervjuer har vi dock ändrat uppfattning och inser att RUP har nog tänkt på det mesta.

Men vi vill påpeka att det krävs att man har goda kunskaper om vad prototyputveckling innebär för att RUP:s direktiv ska kunna utnyttjas till 100%. RUP riktas till systemutvecklare och därför går RUP inte närmare in på att förklara begrepp som prototyputveckling, systemutveckling etc.

Vi upplever prototyputveckling som något positivt inom RUP. Främst p.g.a. att den föreslår prototyputveckling både för användargränssnitt och systemarkitektur. All experimentering är tillåten fram till slutet av elaboration-fasen.

I de efterföljande faserna utvecklar man inte längre prototyper och det är också svaret på frågan: ”När ska man sluta utveckla prototyper?”.

Fram till elaboration-fasen har man främst utvecklat systemet på prototypnivå.
De riktlinjer för prototyputveckling inom ett projekt enligt RUP är:

- evolutionär prototyp vid utveckling av systemarkitektur.
- throw-away (utforskande prototyp enl. RUP) vid utveckling av användargränssnitt.

I slutet av elaboration-fasen kommer man fram till den viktigaste avstämningspunkten i hela projektet, nämligen milstolpen Lifecycle Architecture. Som vi har nämnt är det vid denna milstolpe man stämmer bl.a. av att projektet tekniskt går att genomföra, vilket innebär att bl.a:

- systemarkitekturen mer eller mindre ska vara färdigutvecklad
- att användargränssnittet mer eller mindre ska vara fastställt.

Baserat bl.a. på resultatet av ovanstående avstämningar är det främst upp till kunden att besluta om man ska fortsätta med projektet eller om det ska avbrytas.
I och med att utvecklingen av systemet hittills har skett på prototypnivå har man förhoppningsvis lyckats hålla kostnaderna nere.
Detta innebär att kunden kan välja att avbryta projektet utan att han/hon har förlorat så mycket pengar samt att leverantören, dvs systemutvecklarna, har kvar kundens förtroende.

Sammanfattningsvis visar vår undersökning att prototyputveckling inte är beroende av RUP. Utvecklarna på WPC hade ändå gjort prototyper även utan RUP som systemutvecklingsmetod. RUP, å andra sidan, är beroende av prototyper. Ett exempel kan vara att RUP:s huvudmål är att eliminera risker, vilket ju även är ett av prototypernas främsta syfte. RUP i sig tillför egentligen inget nytt inom detta område utan utnyttjar endast prototyperna på ett effektivt sätt, som vi nämnde ovan; evolutionär prototyp vid utveckling av systemarkitektur och throw-awayprototyp vid utveckling av användargränssnitt.

7 Förslag kring prototyputveckling, RUP samt prototyputveckling i samband med RUP

Baserat på litteraturstudien och den empiriska studien kan vi slutligen komma med följande förslag kring prototyputveckling, RUP samt prototyputveckling i samband med RUP.

Vi har redan tidigare konstaterat att behovet av en bättre definition av begreppet prototyp finns inom WPC. För att undvika framtida missförstånd och underlätta system-utvecklingsarbetet är det viktigt att man i alla fall minst på avdelningsnivå kommer överens om en gemensam definition av begreppet. Som en av utvecklarna föreslår vore det kanske önskvärt att hela Volvo IT kunde enas om en och samma definition, men det kräver en mycket större insats.

Vi har även tidigare konstaterat att man ska utveckla prototyper när man har behov av det. ”*Är kunden och kundens behov kända så behövs oftast ingen prototyp*”, menar bl.a. Burman. Det kanske vore bra att ta reda på kundens behov av prototyper innan man börjar utveckla den. Har han/hon tidigare erfarenhet av att vara kund/beställare? Kan han/hon läsa och förstå en kravspecifikation eller behövs det en prototyp för att hitta alla krav? I så fall, vilken teknik ska användas för att utveckla en passande prototyp?
En annan fråga som kan vara väsentlig att tänka på i sammanhanget; utvecklar man prototyper för kundens skull eller för att man är osäker på sina egna lösningar?

När det gäller RUP så har det framkommit tydligt i vår undersökning att oerfarenhet av RUP och bristen på tid att sätta sig in i RUP gör att RUP känns tungt och otympligt. Som många av de intervjuade påpekade krävs att man arbetar med RUP för att bli bättre på det.
Vi föreslår helt enkelt att man i ett skede som detta, där de flesta är oerfarna RUP-användare, att projektplanen borde anpassas efter utvecklarnas erfarenhet av RUP, samt att man tar de bitar av RUP som passar ett visst projekt. Att dokumentera, tar också mer tid än man tror (vi vet av egen erfarenhet ☺), vilket gör att tid för dokumentation också måste planeras in i tidsplanen från början.
Ett sätt att avgränsa RUP kan kanske vara att man till en början väljer ut några få, men dock viktiga artefakter inom varje roll, för att underlätta upplärningen och användandet av RUP.

För att säkerställa att kunden får det system som han/hon förväntar sig samt att systemutvecklarna har förstått vad som ska utvecklas krävs fler avstämningspunkter, vilket är en synpunkt som Charlie Nordblom starkt poängterade. ”*Det går för snabbt från VAD till HUR”*, som han uttryckte det. Han skulle vilja ha en avstämningspunkt efter brainstorming, en efter framtagningen av use-casescenarierna och en efter utvecklingen av prototypen. Detta stämmer också överens med Runo Burmans bild av hur samarbetet mellan kund och utvecklare borde gå till (se figur 5-2).
Kunden påverkas utifrån och hans/hennes bild av problemområdet förändras ständigt. Det är just vid dessa avstämningspunkter som kunden och systemutvecklaren kan stämma av att de har en gemensam bild av problemområdet och det är det som gör avstämnings-punkterna så viktiga.

8 Förslag till framtida forskning

Inför vår uppsats hade vi många frågor som vi ville besvara. Några av de ursprungliga frågorna lyckades ta sig till vår frågeställning, några var vi tyvärr tvungna att välja bort. När man dessutom sitter och arbetar så intensivt med ett ämne som vi har gjort dyker det alltid upp nya intressanta frågor, som kanske inte direkt har med uppsatsen att göra, vilket gör att man inte kan fördjupa sig i dem. I alla fall inte för den här gången. Istället har vi valt att dela med oss av de frågorna till framtida uppsatsskrivare.

- Vilka ekonomiska för- och nackdelar finns när det gäller prototyputveckling?
- Hur mycket tid och resurser avsätts för att utveckla prototyper?
- Vilka verktyg finns och används idag för prototyputveckling?
  Hur fungerar de?
- Prototyputveckling inom systemutveckling genom tiderna.
  Vad var viktigt då? Vad är viktigt nu?

- Hur ska man på bästa sätt implementera RUP i en grupp av utvecklare som inte tidigare har arbetat enligt RUP?
- Kontrollera och testa RUP:s verktyg för att se hur de fungerar i jämförelse med RUP:s teorier om systemutveckling.
- Jämförelse av RUP i små respektive stora projekt?
- Hur fungerar RUP i virtuella projekt/team?
- Jämförelse av hur RUP fungerar på olika typer av företag?
- Är RUP framgångsfaktorn för lyckade projekt?

- Hur fungerar RUP inom WPC alt. Volvo IT om några år?
  Har det skett några förändringar?

9 Referenser

Litteratur

- Backman, Jarl. (1998). *Rapporter och Uppsatser*. Lund: Studentlitteratur.
- Kruchten, Philippe. (2000). *The Rational Unified Process - an introduction 2nd edition*. Reading, USA: Addison Wesley Longman Inc.
- Löwgren, Jonas. (1993). *Human Computer Interaction*. Lund: Studentlitteratur.
- Mathiassen, Lars., Munk-Madsen, Anders., Nielsen, Peter Axel., Stage, Jan (1998). *Objektorienterad analys och design.* Lund: Studentlitteratur.
- Nationalencyklopedien. (1989). Höganäs: Bra Böcker.
- Preece, Jenny. (1994). *Human Computer Interaction*. Harlow, Storbrittanien: Addison Wesley Longman ltd.
- Sommerville, Ian. (2001). *Software Engineering*. Harlow, Storbritannien: Addison Wesley Ltd.
- Starrin, Bengt., Svensson, Per-Gunnar. (1994). *Kvalitativ metod och vetenskapsteori*. Lund: Studentlitteratur.
- Stevens, Richard., Brook, Peter., Jackson, Ken., Arnold, Stuart. (1998). *Systems Engineering*. Hemel Hempstead, Storbrittanien: Prentice Hall Europe.

Elektronisk media

- Lotsson, Anders. *Rätt ord för bransch och industri*. Internet. Stockholm. IDG: Computer Sweden. 2001-06-01, http://nyheter.idg.se/display.asp?id=010601-cs1.
- RUP Online, Rational Software Corporation, cd-rom, version 2001A.04.00

Föreläsning

- von Dorrien, Christina. *"Analys till design, Prototyper, Evaluering"*, Människa-Dator Interaktion, Datavetenskap, Chalmers Tekniska Högskola/ Göteborgs Universitet, Göteborg 010927.

Viktiga möten på WPC

- Hydén, Ida., Lood, Frank. En introduktion till WPC:s arbete. Göteborg 011001.
- Andersson, Jessika., Johansson, Ulrika: En introduktion till RUP på Volvo IT, Göteborg 010914.

10 Bilagor

10.1 Bilaga 1 - Intervjufrå gor till WPC

1. **Peka ut din roll bland RUP:s workers.**

2. **Vad är din roll i projektet och vad innebär rollen?**

3. **Ger RUP dig en tydlig beskrivning av ditt ansvarsområ de?**
   Om ja
   Om nej - Hur får du annars reda på ditt ansvarsområde?

4. **Har du deltagit i projekt som har tillämpat RUP tidigare?**
   Om ja - Hur många projekt?
   Om nej

5. **Vad har dina erfarenheter av RUP varit hittills?**
   Vad anser du vara fördelarna med RUP?
   Vad anser du vara nackdelarna med RUP?
   Andra kommentarer?

6. **Hur klargör du otydliga frå gor som dyker upp under ditt arbete?**
   T ex otydliga/nya krav

7. **Hur kommunicerar ni i projektet?**

8. **Vad är syftet med prototyputveckling för dig?**

9. **Använder/utvecklar du nå gon prototyp för att underlätta ditt arbete i projektet?**
   Om ja - Hur utvecklar du i så fall prototypen?
   Om nej - Varför inte? Finns inte behovet?

10. **Vad anser du vara fördelar med prototyputveckling?**

11. **Vad anser du vara nackdelar med prototyputveckling?**

12. **Skulle du kunna utföra dina arbetsuppgifter utan prototyputveckling?**

13. **På verkar RUP på nå got sätt prototypsutvecklingen?**

14. **Använder du nå gon standard vid prototyputveckling?**

15. **Har du behov av en standard?**

16. **Vad upptar mest av din tid i projektet?**

17. **Har du nå gra förslag kring prototyputveckling/användning?**

10.2 Bilaga 2 - Sammanställning av de tolv intervjuerna på WPC

	Fråga/Person	Person 1	Person 2	Person 3
	Roll	**Arkitekt**	**System Analyst**	**Designer,** Arkitekt**,** DB Designer, Implementer, Testare
2	**Har du tidigare deltagit i projekt som har tillämpat RUP?**	Nej (1)	Ja (3)	Ja (minst 2)
3	**Vad är dina erfarenheter av RUP?**	RUP är inte gjord för denna typen av projekt. D.v.s vi gick in i projekt utan att veta vad vi skulle göra egentligen, men vi hade en deadline, som dessutom låg nära i tiden. Då är RUP som process inte stödjande i det läget utan mer en tyngd som man måste släpa med sig. Jag antar att när man har kört RUP tillräckligt många gånger, så kommer det att leda till att vi kommer att få 2-3 varianter av RUP, baserade på projektstorlek.	• Fördelen är att man inte bara koncentrerar sig på att verkställa projektet. Man får möjligheten att tänka igenom redan i den inledande fasen: ”Vad är problematiken?”, ”Hur ska det lösas?” • Möjligheten att integrera olika parter/ intressenter i processen. (kund, leverantör). • Att man har tydliga roller. Var och en vet vad ens ansvar eller uppgifter är. Det är strukturerat sätt att arbeta på. • Har man 19 usecase så ser jag inte något annat sätt att hantera krav och information på ett bättre sätt än att använda en välutvecklad metod. • Nackdelen är att RUP är så pass omfattande att man måste ta den med måtta för att det inte ska bli alltför betungande.	• Bra i början av projektet när man ska ta fram use-casen, för att ta reda på vad man ska göra. Man börjar och jobbar sig in i det. • Nackdelen är att man måste gå igenom ett antal faser, men för det krävs tid också. Ibland är projekten så korta att man inte får tillräckligt med tid i alla faserna.
4	**Hur kommunicerar ni i projektet**	• Projektmöten varje morgon. • Två dokumentsajter, en intern och en extern som man använder tillsammans med kunden.	Jag kommunicerar mest via textdokument, samt skisser/ritningar som vi för över till PowerPoint och sprider till berörda. ”En bild säger mer än tusen ord”. Det är väldigt bra att göra en bild av en text.	• Morgonmöte, organiserade möten, spontana samtal. • Dokumentation på gemensam projektdisk. • PowerPoint-bilder och whiteboard.

5	**Vad är syftet med prototyputveckling för dig?**	• Visualisera och visa upp för någon som inte har varit med i diskussionen tidigare. • Ett sätt för att få kunden att lättare förstå vissa problemställningar.	• Avbildning av krav, eventuella lösningar och lösningsförslag. • Tydliggöra och förklara för alla inblandade i projektet. Textdokument svarar på frågan VAD. Bilder och prototyper svarar på frågan HUR.	• Man bygger prototyper för ett visst usecase. För att testa saker som man inte vet hur de kommer att fungera. Sedan slängs prototypen. • En annan variant är att prototypen blir den färdiga produkten.
6	**Använder/utvecklar du prototyper i detta projekt?**	Nej, är inte inblandad just i den biten. DB-prototyping: För att testa att vissa saker fungerar överhuvudtaget. Throwaway.	Inte just jag, men man gör det i projektet.	• Ja, jag har byggt en förenklad applikation, som innehåller rätt mycket av det som kommer att finnas i den riktiga applikationen. • För att visa kunden och för att testa.
7	**Fördelar med prototyputveckling?**	• Visualisera något som kan vara svårt att läsa sig till i ett dokument. • Lättare att förstå om man ser det bildmässigt. Lättare att få fram en idé på det sätttet.	Kunden och användarna får det lättare att förstå hur man ska lösa problemen. Hur man ska interagera med systemet och hur den tänkta lösningen kommer att se ut. Det är först när man har levererat storyboarden till kunden som det förstår innebörden av alla ord de har läst. Då kan de komma med åsikter och nya idéer.	• Man hittar brister tidigt i processen. • Man kan visa ett prototypat use-case för kunden, vilket gör att de blir mer positivt inställda till projektet.
8	**Nackdelar med prototyputveckling**	Beroende på hur man utför prototypen. Ibland kan den uppfattas som det slutliga resultatet, av dem som man visar den för.	• För kort tid på sig i projektet att man inte hinner göra dem. • Kunden är inte villig att betala för prototypen, men då gäller det att ”missionera”, att få kunden att inse att den byggs för dem.	• P.g.a. tidsbrist finns det risker att prototypen blir det slutliga systemet. • Det får inte bli för mycket prototyper.
9	**Skulle du kunna utföra dina arb.uppg utan prototyputveckling?**	Antagligen inte. Man gör alltid någon form av prototyp. Så fort man visar något visuellt för att visa problem så att är det en prototyp.	Jag skulle kunna klara mig utan i mitt jobb, men det skulle inte vara fördelaktigt för projektet. Det är andra som tar över mina usecases och utvecklar prototyper.	Nej, man bygger alltid en form av prototyp.
10	**Påverkar RUP på något sätt prototyputv?**	Inte som jag ser det, men RUP har ett sätt att se på prototyper. Där det finns möjligheten att ha prototypen och använda den som ett underlag att gå vidare med.	Det fungerar bra i RUP om man har de rätta förutsättningarna, d.v.s gott om tid, resurser och ekonomiska förutsättningar.	Ja, när man har kommit så långt att man har gjort use-casen och kanske börjat lite med analys och design kan man styra prototypen på något sätt. Typ i mitten av elaborationfasen.
11	**Använder du någon standard**	-	-	Nej

12	**Har du behov av en standard**	-	Det hade varit bra. Frågan är om det går att dela in projekt i olika grupper och säga att den för denna projektgruppen ska man använda den och den prototypen. Det kanske går.	Nej. Det finns säkert någon underförstådd standard. Ibland gör vi storyboard, det kan vara en standard.

	Fråga/Person	Person 4	Person 5	Person 6
	Roll	**Arkitekt,** Designer, Implementer	**Designer** Implementer	**Implementer**
2	**Har du tidigare deltagit i projekt som har tillämpat RUP?**	Ja, minst (1)	Ja. (2)	Nej. (1)
3	**Vad är dina erfarenheter av RUP?**	• Med RUP är det ganska mycket pyssel runt om. Rose är ett stort, klumpigt verktyg som har buggar och inte är heltäckande. • RUP känns mer naturligt för ett stort projekt som sträcker sig över flera år med många människor inblandade. • RUP är väldigt mycket dokument.	• Det är ett sätt att dokumentera det man tänker och det man pratar om. • Utvecklingsarbetet och ordningen på koden blir mycket bättre. • Den stora nyttan med RUP i detta projektet är att vi har insett att det är riktigt bra. • Nackdelen är att det tar så lång tid och att vi är ovana med det. Man måste få tid att lära sig och det har vi inte riktigt haft tid att göra.	• Det är bra att styra upp saker och ting på. • Få kunder och oss själva att tänka efter lite före. • Sen kan det vara lite trögt samt att vi har begränsad erfarenhet av det på avdelningen.
4	**Hur kommunicerar ni i projektet**	• Dokument • Möten • Diagram	• Möten för allmän info. • Dokument för krav mm på ett gemensamt ställe. • Whiteboard när man ritar idéer och lösningar. Dessa idéer skissas ner för eget bruk.	Möten till större delen (80%). Resterande del Word- och Excel-dokument.
5	**Vad är syftet med prototyputveckling för dig**	• En prototyp är bara något som inte ger sken av att vara den slutliga lösningen. Man tar fram ett mellanting för att ha något att visa och diskutera kring. • Det ska gå fort, ge snabbt resultat för att stämmas av mot kund. Det kan var en pappersgrej, men också klickbar variant.	• Testa funktionalitet och sina idéer. • Komma fram till hur man ska utveckla på ett smart sätt.	• Försöka att få kunden att förstå. • För utvecklarna att testa teknik.
6	**Använder/utvecklar du prototyper i detta projekt?**	Jag gör en databas-modell, funderar på hur den ska se ut och sedan bygger jag en prototyp kring den. Sedan stämmer jag av den mot kund och dokumentation och folk.	Ja. Utifrån use-caset utvecklades en kodprototyp och från den återanvändes kod till det slutliga systemet	Nej, inte i detta projektet.

		En iteration där prototypen blir bättre och bättre och tillslut blir det fulla systemet. Det är en variant av evolutionär där slutmålet är systemet, men med vissa inslag av incrementell eftersom det är en form av frysning när de rensar i koden. All onödig kod ska vara borta, felaktig kod ska vara rättad och datamodellen ska se riktig ut utifrån det vi vet då.		
7	**Fördelar med prototyputveckling?**	• Ett billigt sätt att ta fram något fort så att man själv kan se att tekniken håller. • Att man har fattat vad kunden vill ha och att kunden har fattat vad den vill ha. • Kommunikation mellan kund och utvecklare.	Testa sina idéer och strukturera upp koden.	• Man får kunden att förstå och för att testa teknik. • Himla bra i kommunikation med kund.
8	**Nackdelar med prototyputveckling**	• Ser ingen direkt nackdel med prototyp-utveckling, men det beror på vem som prototypar. • Man måste bestämma sig innan: Ska vi bygga för att slänga eller ska det bli det slutliga systemet. Ska man iterera måste man gå in och städa vid varje iterationsstopp. • Risk att kund uppfattar prototypen som det slutliga systemet. Man får förklara att prototypen är ett skyltfönster. ”Det ser läckert ut, men du vet att det måste finnas något bakom också.”	• Risken är att man drar det lite väl långt, pillar på användargränssnittet mest för att man tycker att det är kul. • Risk att man för tidigt låser in sig på en viss lösning på ett problem. • Risk för att prototypen blir det slutliga systemet p.g.a överarbetning. • •	• Svårt för kunden att förstå att det är prototyp, att allt inte fungerar. • Man kan låsa sig för tidigt vid en viss lösning. • Man måste lämna pappersstadiet för att få svar på frågorna. Att bara komma med papper räcker inte. Det är först när de ser att det rör på sig som man får en reaktion.
9	**Skulle du kunna utföra dina arb.uppg utan prototyp-utveckling?**	-	Det har jag svårt att tro. Jag är mer praktisk och provar och provar, frågar någon och man inte har löst det kanske man till slut läser i en bok.	Ja.
10	**Påverkar RUP på något sätt prototyputv?**	• Prototyping utan RUP går fortare än prototyping med RUP, om man inte är väldigt duktig på RUP.	Ja, det skulle det kunna göra. I och med att RUP förespråkar iterationer så blir det automatiskt att man gör någon form av prototyp.	Det är lite mer uppstyrt, men mest påverkar det negativt, i och med att så många inte kan. Man måste jobba med RUP för att bli bättre på det.

		• En prototyp ska göras snabbt, då måste man vara duktig på RUP och Rose för att ha någon glädje av det. • Ju större projekt desto större nytta. RUP känns mer naturligt för ett stort projekt som sträcker sig över flera år med många människor inblandade. • RUP är väldigt mycket dokument. Ska man ha RUP för prototyping ska man ha en nerbantad variant och man måste ha folk som kan RUP. Det är en uppstartsträcka på ett antal veckor och ibland månader.		
11	**Använder du någon standard**	-	För mig är det så fort jag testar något så är det en prototyp. När man säger prototyp så ser folk i regel en HTML-prototyp. De flesta inte ser vår utvecklingstestning som prototyping, men jag ser det som det.	-
12	**Har du behov av en standard**	-	Standard vet jag inte om det behövs, men utbildning och upplysning.	-

	Fråga/Person	Person 7	Person 8	Person 9
	Roll	**Arkitekt,** System Analyst, Designer DB Designer	**Implementer**	**Stakeholder,** Kund, Kravställare, godkänner kravbild, godkänner lösnförslag
2	**Har du tidigare deltagit i projekt som har tillämpat RUP?**	Ja, (3)	Nej (2)	Ja, 2(3)
3	**Vad är dina erfarenheter av RUP?**	• Utvecklingsmässigt tror jag att RUP är bra för analysfasen och även för konstruktionsfasen. Att jobba use-casebaserat är positivt. • RUP är ett bra styrmedel för att få projekten att gå i hamn. • Nackdelar: Längre startsträckor i projekten dels för att RUP är nytt för många. Man får ta ut längre projekt-tider. Högre kostnad för projekten. • Alla accepterar inte att vi kör RUP. De kör på som förut. Det kan vara svårt för projektledning att följa upp vad de gör egentligen.	• RUP är ett annat sätt att säga samma saker. Iterativ utveckling är ju inget nytt. Men det här gör man kanske för att få ner det på papper, vad vet jag? • RUP är helt ok om man går på idén och inte stirrar sig blind på alla punkter. • Nackdel: om man stirrar sig blind på det och låter RUP styra framför det sunda förnuftet.	• Ett strukturerat sätt att ta fram kommunikationsmodeller med kunden. • Ett strukturerat sätt att förklara vad man skall göra under hela projektet och hur man kommer att göra det. • Nackdel: Komplext och svårt. Man måste göra en skalning av metoden och bestämma sig för vilka delar man använder. • Svårt att förstå hur att delarna hänger ihop.Vissa diagram är komplexa och visas därför inte för kunden, men man förklarar heller inte att de finns. • Inte hittat koppling till prototyper
4	**Hur kommunicerar ni i projektet**	• Telefonkonferenser, resor, projektmöten. • På möten används white-board och projektor. Sekreterare som för protokoll. • Info mot kund läggs i en LotusNotes-db, som de kan nå via webben. • HTML-prototyp för funktionalitet. • Word- PowerPoint-, Excel-dokument. • Versionhanteringssystem. • Mail	• Kommunikationen borde ske genom dokument, men eftersom tid inte finns att läsa dem, frågar man i första hand andra. Hittar man inget svar går man till slut till dokumenten. • Kommunikation sker genom team-leadern som får sin info från PL.	• Inplanerade workshops med metodstöd som skrev det vi sa i usecases. • Skissa på whiteboard, tecknat ner och skickat ut via mail för att bekräfta att man hade tänkt rätt. • Ibland mötesanteckningar. Ibland egna anteckningar. • Storyboard för att visa flödet i verktyget. Designprototyp för att visa användargränssnitt.

5	**Vad är syftet med prototyputveckling för dig**	• Två syften. • Den ena utvecklas för att till slut bli slutprodukten. • Den andra, HTML- prototypen, utvecklas för att få förståelse av use-casen mot kunden samt som diskussionsunderlag med kund, men även en del av utvecklingsarbetet där HTML-prototypen ”lyfts in” i serverkoden och statisk information ersätts med dynamisk genererad information.	• Testa olika saker, framför allt riskområden, saker man är orolig för att de inte ska fungera. • För att ta fram användargränssnitt och visa kunden att man faktiskt gör något.	Få in kommentarer, förslag och lösn-ingar från mina kunder. Genom att visa något och be om kundernas input skapar man engagemang och ägandeskap hos kunden. Jag tar dem som en slags ”gisslan”med storyboarden. De kan inte protestera i slutet för de har själva varit med att bestämma.
6	**Använder/utvecklar du prototyper i detta projekt?**	Ja, jag gör den körbara arkitekturen. Ingen användargränssnittsprototyp, utan en inriktad mot serverdelarna. Vi bygger upp en miniproduktionsmiljö så att det blir så billigt som möjligt, men ändå avspeglar produktionsmiljön.	En HTML-prototyp utvecklas i projektet. Jag själv utvecklar t.ex. en viss komponent som jag prototypar och stoppar in i HTML-prototypen för att se om den fungerar.	Tidigt i projektet gjorde jag olika PowerPoint-presentationer för att förklara flödet. Jag jobbar mycket med bildspråk och ritar bilder och modeller för att förklara. Whiteboard-skisser för att förklara min mentala modell över hur en lösning ska fungera.
7	**Fördelar med prototyputveckling?**	• Tidig känsla för var man problemen i systemen och vad man ska lägga mest tid på, vilka risker ska elimineras. • Få kunden att förstå. De måste ha något riktigt att se på.	• Testa olika saker, framförallt riskområden. • För att ta fram användargränssnittet och visa kunden att man gör något.	• Jag kan ge användarna ”en glimt av framtiden”, d.v.s. hur systemet kommer att se ut och fungera. Det kan vara svårt för kunden att förstå framtida lösningar bara beskrivna i ord. • Jag kan på ett bättre sätt be om deras input i systutvecklingsprocessen. För att kunden ska känna att det är deras lösning. • Sälja in verktyget till andra i organisationen. • Utbildning.

8	**Nackdelar med prototyputveckling**	• Risk att man låser sig för tidigt vid en lösning i processen. • Risk att kund t.ex. tittar på färger när man själv egentligen vill visa funktionalitet och tvärtom.	Ser inga nackdelar	• Om man visar prototyp för tidigt kan man få negativ feedback om man inte gör och visar förbättringar utifrån feedback. Negativ inställning kan påverka arbetet framöver. • Man kan lova för mycket med en prototyp. • Låser fast sig i en idé. • Olika yrkesgrupper fokuserar på olika saker.
9	**Skulle du kunna utföra dina arb.uppg utan prototyp-utveckling?**	Mycket svårt arbetssätt. Det är inte naturligt. Även utan RUP gör man det.	Ja, det skulle jag nog, men jag har svårt att tänka mig det. Jag hade nog gjort en själv i min egen lilla skala. Så har man alltid något att utgå ifrån ifall man vill ändra något.	Jag är ganska beroende av prototyper för att visa mina kunder vad det är jag jobbar med och hur jag får fram deras önskemål. Hade jag inte fått en prototyp från WPC hade jag ändå gjort en själv
10	**Påverkar RUP på något sätt prototyputv?**	Positivt att det byggs in i det här med körbar arkitektur och releaser efter varje iteration. Innan RUP var det inte så releasorienterat. Nu vet man i varje iteration vad som ska göras och vad nästa release ska innehålla.	Det har jag inte tänkt på en gång. Det är möjligt att arbetsuppgifterna hade sett annorlunda ut utan RUP. Jag kan inte avgöra det.	Nej, jag tycker inte det. Jag ser ingen riktig koppling.
11	**Använder du någon standard?**	För HTML-prototyp har vi utarbetat en egen slags standard, men jag vet inte om det finns något nerskrivet.	Nej, inte för prototypen i sig.	Skisser (PowerPoint och papper), whiteboard
12	**Har du behov av en standard?**	Jag tror att det vore bra om vi internt hade riktlinjer för det. Lite mer guidelines. Att inom VIT ha en gemensam syn på prototyper vore en fördel.	Ja, det gör jag när det gäller begrepps-förvirringen. Jag är mycket för standarder överhuvudtaget. Alla kanske har olika uppfattning om vad en prototyp är för något.	Ur kundperspektivet vore det bra att veta vilka prototyp man får och i vilket format de är. Viktigt i vilket format den är i. Det är stor skillnad mellan en prototyp i PowerPoint, som visserligen är klickbar, och en som är gjord i ett verktyg och är som en första färdig lösning. Alla förstår inte skillnaden mellan en HTML-prototyp och en klickbar PowerPoint-bild.

	Fråga/Person	Person 10	Person 11	Person 12
	Roll	**Project Manager**	**User interface designer,** Team leader, Designkoordinator	Möjligtvis **stake-holder,** men har ingen rupdefinierad roll. Gruppledare
2	**Har du tidigare deltagit i projekt som har tillämpat RUP?**	Ja, en gång tidigare (2)	Ja. (3)	Ja. (3)
3	**Vad är dina erfarenheter av RUP?**	• Fungerar ganska bra. Problemet är att vi inte är vana att jobba modellariserat. • Stor kontrast mot att inte köra något alls, som vi gjorde tidigare. • Stor lärotid. • Extraarbete som man inte har räknat med tidsmässigt. Viktigt att i framtiden lägga in mer tid för sådana aktiviteter.	Märker inte av det i mitt arbete. Fördelen är väl att det strukturerar upp projektet på ett bra sätt, men jag vet inte om något annat är bättre eller sämre.	• Det är en metod som är beskriven. • Man har hittat på egna ord med flit för att man inte ska ta med dig gamla värderingar. Vi tolkar RUP på vårt sätt. • RUP definierar rätt saker. • Samtidigt är det lite för luddigt. Svårt att veta när man går igenom de olika faserna och vad man ska göra i dem. • RUP är en stor kompromiss mellan olika utvecklings-metoder. • RUP är så stort, vilket gör att det blir svårare.
4	**Hur kommunicerar ni i projektet**	• Teammöten 2ggr/vecka. Teamen har sen möten i sina team. • Info skickas ut via mail. • Helst hade jag velat ge ut ett veckobrev. • Även PowerPoint-bilder och prototyper	• Morgonmöten varje morgon i flera veckors tid. Nu kör vi 2 möten/vecka. • Gemensam server med dokumentation.	• Workshops. • Man sitter ihop och diskuterar. Sen för man ner det på några modeller och för in det i dokument, som man kan använda för kommunikation. • Mycket av kommunikation är genom dokumenten och det muntliga.
5	**Vad är syftet med prototyputveckling för dig**	Att stämma av med kunden så att de vet vad de får innan det börjar kosta för mycket.	• Ett som WPC jobbar på. Det ingår i processerna. • Ger kunden en visuell bild av vad han egentligen kommer att få. Kunden har fått se och fått tänka till både en och två gånger. • Upptäcka problem på ett tidigt stadium innan den färdiga produkten.	• Testa utan teknik. • Lära sig. • För att få en gemensam bild mellan kund och leverantör av det man ska göra och hur slutprodukten kommer att se ut. • För att få krav och komma överens med kund vad du ska leverera.

6	**Använder/utvecklar du prototyper i detta projekt?**	• Vi gjorde en prototyp, men den var till för kunden gentemot sina kunder. Det hade varit bättre om vi hade haft en prototyp för oss mot vår kund. • Vi gjorde storyboards, men de var inte tillräckligt heltäckande och för avancerade. • Designskisser för användargränssnittet. • ”Usecase är en prototyp fast beskriven i ord”	Ja, HTML-prototyp.	Inte jag, men i projektet använder vi HTML-prototyperna ganska mycket med ganska lyckat resultat. Främst HTML-prototyper och en del PowerPoint-prototyp också. Prototypen blir det man levererar. Vi slänger den inte utan vi gör en prototyp som blir den slutliga produkten. Hålla skillnad på användargränssnitts-utvecklare och programmerare, så att de jobbar med det som de är bäst på.
7	**Fördelar med prototyputveckling?**	• Stämma av exakt hur det ska se ut innan du ska utveckla det. • Att stämma av innan produktionssättning.	• För att kunna se visuellt. När det är tajt om tid kan det vara bra att ha en prototyp.	• Man kommer överens om kraven och att man får se hur det kommer att se ut. • Relativt billigt. Att göra fel i HTML-kod är billigare än att göra fel i applikationskod. • Kan användas för demonstrationer, utbildning och för testning.
8	**Nackdelar med prototyputveckling**	Kan vara kostsamt. Gör du det i HTML och javascript så kan det bli dyrare än om man jämför med storyboard och designskisser.	• Kunden tror ofta att det är den färdiga produkten och förstår inte varför projektet kommer att dröja några månader till. • Kunden hakar upp sig på det som inte är väsentligt för avstämningstillfället, t.ex. fel färger. • De vill inte alltid betala för det.	• Det tar tid. • Man kan inte utveckla en prototyp bara för att slänga. Att slänga en prototyp och börja om från början kostar alldeles för mycket.
9	**Skulle du kunna utföra dina arb.uppg utan prototyputveckling?**	Ja	Ja, resultatet HTML-mässigt skulle bli detsamma. Men det underlättar för mig också att se ett färdigt. Hade vi inte gjort en HTML-prototyp hade vi säkerligen gjort någon form av flödesprototypgrej för oss själva för att se sammanhanget.	Jag tror att det är svårt utan prototyper men rent tekniskt skulle det gå. Om man har väldigt mogna kunder behöver man inte prototyper.
10	**Påverkar RUP på något sätt prototyputv?**	Det förenklar väl. När du har use-casen och så sånt så är det mycket lättare.	Nej, inte konkret. Det har det säkert eftersom önskemålen kommer från projektledning och högre upp.	RUP säger att man gärna kan jobba med prototyper, främst i elaborationfasen. Det hör ganska naturligt till elabfasen eftersom det är där man testar och experimenterar.

11	**Använder du någon standard**	När vi säger prototyp så menar vi HTML-prototyp.	Inte standard allmänt. • HTML är min standard, men jag har inget begrepp. • Många förvillar sig med att säga prototyp eller demo. Ingen vet riktigt egentligen.	Nej, inte direkt.
12	**Har du behov av en standard**	Är det en HTML-prototyp så är det ingen större förvirring. Är det en PowerPoint-prototyp så vet man också vad det är.	Ja, i alla fall när det gäller att skilja åt prototyp och demo. Man säger samma sak men man menar olika.	Kanske inte en standard, men en bättre definition