


Handelshögskolan
VID GÖTEBORGS UNIVERSITET
Institutionen för informatik

2004-05-26

Systemarkitekturell signifikans och problematik i Rational Unified Process

Abstrakt

Uppsatsen behandlar problematik som uppstår då arbetet med systemarkitekturella frågor inte tillåts ta den plats som är nödvändig inom en modern systemutvecklingsprocess för att infria arkitekturens positiva egenskaper, exempelvis produktkvalitet och hållbarhet. Processen som undersökts i samband med begreppet systemarkitektur och dess innebörder är Rational Unified Process (RUP) då den är arkitekturcentrisk. Arbetet lyfter fram viktiga aspekter och beståndsdelar inom RUP vars utförande direkt bidrar till utvecklingen av en god systemarkitektur samt att förklarar och motiverar dem. Resultatet av uppsatsarbetet pekar slutligen på att arkitektur måste ta mer plats inom utvecklingsprocessen än idag men det tillåts enbart om kunden inser de fördelar en systemarkitektur för med sig då det är kunden som betalar för projekt och dess resurser. Vidare påvisar uppsatsen som resultat ett behov av en ny arkitekturupplysande beståndsdel i RUP. Uppsatsen är främst en kvalitativ litteraturstudie där teori och principer företräder empiri och har arbetats fram med stöd av sakkunniga i näringslivet.

Nyckelord: RUP, Rational Unified Process, Systemarkitektur, FURPS+

Författare: Carl Strömhielm
Handledare: Wera Tegner Johansson
Magisteruppsats, 20 poäng

Förord

I förordet vill jag tacka de personer som på olika sätt hjälpt till med att färdigställa uppsatsen genom att generöst dela med sig av sin tid och kunskap samt visat intresse för mitt arbete.

Speciellt tack till:

Sigvard Svensson, metodexpert, AcandoFrontec AB
Mats Wessberg, Consolidate Sweden AB

Innehållsförteckning

1. Inledning

1.1. Introduktion och bakgrund

Det problem uppsatsen behandlar är av allmän karaktär på så vis att det inom systemutvecklingen är ett allmänt erkänt problem att utvecklingen av systemarkitektur i programvaruutvecklingsprojekt generellt sett inte ges tillräckligt med utrymme för att dess positiva aspekter ska realiseras i önskad grad. Det negativa resultatet som kommer sig av att systemarkitekturen inte ges tillräckligt utrymme i en utvecklingsprocess är huvudsakligen förlorad produktkvalitet överlag samt reducerad livslängd av den färdigställda produkten då den i sig kan bli svår att vidareutveckla. Problematiken kan visa sig på ett flertal nivåer i systemet varav ett exempel kan vara svårigheter att införliva ny teknik i systemet. De kvalitativa fördelar som riskerar att gå förlorade i ett projekt där arkitekturella frågor inte får tillräckligt utrymme har en väsentlig inverkan på den slutprodukt som levereras till kunden av utvecklarna vilka kunderna inte är tillräckligt medvetna om. Den bristande insikten från kundens sida om arkitekturens vikt för den beställda produkten får till följd att de resurser som ställs till projektets förfogande inte fokuseras på arkitekturfrågor. I fokus för kundens intresse står främst produktens funktionalitet vilket ses som det primära kriteriet för huruvida den slutligen levererade produkten lyckades uppfylla ställda förväntningar eller inte. Vad som negligeras i sammanhanget är det direkta beroendeförhållande som ett systems funktionalitet har till systemarkitekturen. Uppsatsen vill belysa problematik som uppstår härav samt svara på frågeställningar vars svar kan erbjuda en möjlig lösning av problemet.

Bakgrunden till problematiken som lett fram till ett ökat, men än inte tillräckligt, fokus på explicita arkitekturfrågor kan besvaras genom en evolutionär syn på systemvetenskapen och dess tillämpningsområde. Bland annat har systemkraven ständigt ökat och arbetet med att uppfylla dessa krav har lett till en alltgenom ökande komplexitet för såväl resultat som underliggande utvecklingsmiljöer och processer. En allmänt rådande uppfattning är att systemutvecklingen gått från stabila miljöer, hårt integrerade produkter och vattenfallsmetoder till komplexa och föränderliga miljöer, produkter och iterativa utvecklingsprocesser. Utvecklingen mot ökande komplexitet har fört med sig att tidigare utvecklingsmetoder blivit omoderna och antingen behövt omarbetas drastiskt eller att helt nya utvecklingsmetoder tagits fram. I takt med att nya utvecklingsmetoder och processer arbetats fram har man behållit vissa delar och synsätt från tidigare men också utvecklat nya idéer och koncept. Framväxten av systemarkitektur som en uttalad systemutvecklingsföreteelse är ett exempel på vad som inte haft en explicit plats i systemutvecklingen. Behovet av arkitektur har exempelvis synliggjorts genom ökande teknisk och systemär komplexitet. Läsning av både fackpress, litteratur och samtal med sakkunniga påvisar också arkitekturbegreppets aktualitet i dagsläget som följd av ökad komplexitet och ökade systemkrav. Begreppets mer frekventa förekomst tyder på ett växande problemområde och en ökad uppmärksamhet.

De ämnesområden som behandlas i uppsatsen är systemutvecklingsprocessen Rational Unified Process (RUP) och systemarkitektur vilka valts bland annat för sitt ömsesidiga beroendeförhållande.

Kortfattat kan RUP introduceras som en modern systemutvecklingsprocess av heltäckande slag då den syftar till att beskriva en produkts utveckling från dess mest initiala stadium till levererad produkt. Processen är uppbyggd kring ett antal så kallade bästa praxis och nyckelegenskaper som är grundläggande för dess utformning och identitet. Bland dem kan här nämnas krav- och riskdriven samt arkitekturcentrisk systemutveckling.

Sammanfattat kan systemarkitektur sägas organisera systemutvecklingsprocessens resultat på ett förnuftigt sätt och är en starkt bidragande faktor till att färdigställa ett lyckat projekt. Arkitekturens betydelse för systemet, oberoende om det är IS[1], IT[2] eller IS och IT tillsammans är den en förutsättning för att systemet ska uppfylla initiala krav men samtidigt klara av framtida förändringar.

1.2. Syfte

Syftet med uppsatsarbetet har varit att genom analys av problemområdet tillsammans med dess största ingående beståndsdelar, RUP och systemarkitekturer, eftersträva framläggandet av en grund eller alternativt ett relevant resonemang som bidragande faktor till problemets lösning. I övrigt eftersträvas också att erbjuda läsaren ett holistiskt betraktelseperspektiv av de två ämnesområdena för att ingjuta förståelse för hur de hänger ihop, hur de kompletterar och eventuellt substituerar varandra samt vilka implikationer de medför för varandra och systemutvecklingen som företeelse. Av denna anledning berör uppsatsen också några för objektorienterade systemutvecklingsmetoder grundläggande praxis och en redogörelse för begreppet systemarkitektur i allmänhet och inom RUP i synnerhet. Anledningen till ovan vald ambition är åsikten om att det är för alla inblandade parter ett beprövat och bra sätt att producera ett intressant material på som i slutänden ämnar svara på vald frågeställning samt därmed även uppfylla uppsatsarbetets syfte. Det ligger även inom ramarna för uppsatsarbetet att på ett förkunnande sätt omfatta ämnesområdena för att åskådliggöra erhållna insikter och kunskaper.
Inom ramarna för vad som skrivits under rubriken introduktion och bakgrund ämnar uppsatsen att genom besvarandet av sina frågeställningar bidra till att belysa en inom systemutvecklingen som företeelse allmänt utbredd problematik angående otillräckligt fokus på systemarkitekturella frågor. I sin utformning vill uppsatsen påskina hur denna problematik har uppkommit samt hur den kan komma att åtgärdas. För ändamålet har systemutvecklingsprocessen RUP valts eftersom den är en arkitekturcentrisk process och en i näringslivet väl spridd och allmänt högt ansedd produkt. Uppsatsen vill också redogöra för vad en systemarkitektur är i allmänhet såväl som inom RUP i synnerhet.

[1] Informationssystem och
[2] Informationsteknologi ses som åtskiljda saker. I texten åsyftas att det finns både hårdvarurelaterad- såväl som mjukvaru- och systemrelaterade arkitekturer.

1.3. Avgränsning

Uppsatsarbetet har inte för syfte att erbjuda någon teknisk eller teknologisk genomgång och framställning av respektive ämnesområden eller problemområde. Arbetet behandlar sagda ämnen och problem ur ett holistiskt och akademiskt perspektiv framför ett mer specifikt och detaljorienterat metod-, teknik- och teknologiorienterat perspektiv. Övergripande principer, ramverk och teorier för tillämpning går före empiri och praktik. Praktiska avgränsningar av undersökningsområdet har gjorts utifrån de undersökta ämnenas inbördes struktur och relation till varandra. Tidigare kunskap och insikt om RUP har lett till att fokus på processen anpassats efter begreppet systemarkitekturs innebörd eftersom detta ämne, om än av central betydelse för RUP, endast aktivt behandlas i vissa delar av RUP. Främsta anledningen till att avgränsning gjorts på sagda vis var sättet de båda ämnesområdena berörde varandra och att kopplingen mellan dem var som starkast i det valda undersökningsområdet som kretsar kring analys och design med visst initialt fokus på krav. Även insikt om att ett övergripande allmänfokus på båda ämnena i sin helhet i grund hade lett till en, för uppsatsen, allt för stor, omfattande och tung uppgift spelade också in i avgränsningen. Utan en aktiv avgränsning av undersökningsområdet hade arbetet inte lett fram till någon slutgiltig frågeställning eller problem att utreda utan enbart resulterat i en allmänt redogörande uppsats utan akademiskt intresse. Mycket viktigt att påpeka är också att uppsatsen enbart syftar till att behandla utredning av problemområdet och dess beståndsdelar utifrån ett nyutvecklingsperspektiv vilket har stor betydelse för hur man arbetar med arkitekturen. I fall där det är fråga om vidareutveckling av redan befintliga system är också arkitekturen i många fall befintlig.
Problemområdet kretsar kring krav, analys och designförfarandet inom systemutvecklingsprocessen RUP och främst de delar där arkitekturbegreppet har en direkt och aktiv inverkan, betydelse eller relevans. Vad gäller arkitekturbegreppet har avgränsning gjorts där begreppet övergår i alltför tekniska perspektiv som inom Software Engineering där explicita programvarutekniska och implementeringsmässiga aspekter inte berörts.

1.4. Frågeställningar

Den huvudfrågeställning som kommit fram är resultatet av ett omfattande men grundläggande undersökningsarbete av områdena RUP och systemarkitekturer både var för sig och tillsammans. Det problemområde som uppdagats och bildat utgångspunkt för ett mer specifikt problem ses vara av en allmänt utbredd och betydande karaktär. Den frågeställning som tett sig bäst lämpad att utreda är grundad på studiernas utveckling samt betydande samtal med sakkunniga är:

1. **På vilket sätt kan det tydligare framgå att ett färdigt projekt inte enbart överlämnar produktfunktionalitet till beställaren utan också levererar en systemarkitektur då det motiverar arkitekurella frågors relevans för beställaren?**

Delfrågor som på vägen visat sig lämpliga att besvara för att i slutligen kunna besvara huvudfrågeställningen är:

1. **Varför är det viktigt att bli tydligare med arkitekturen samt vari ligger huvudfrågans bakgrund?**

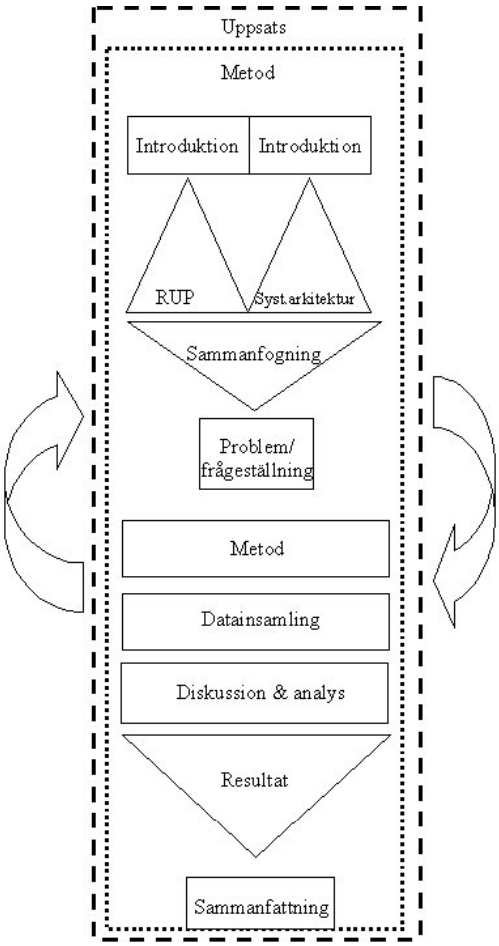
2. **Vad är en systemarkitektur i allmänhet och i synnerhet i anknytning till RUP?**

3. **Vad gör systemarkitekturen betydande och vilka implikationer har den för systemutvecklingsprocessen?**

4. **Hur arbetas en systemarkitektur fram inom ett RUP-baserat utvecklingsprojekt?**

1.5. Disposition

De två olika områdena, RUP och systemarkitektur, introduceras på ett grundläggande sätt ur ett så neutralt perspektiv som möjligt. Områdena presenteras i den omfattning det kan tänkas nödvändigt för berörda läsares förståelse. Fokus läggs på hur systemutvecklingsprocessen RUP är förbunden med arkitekturbegreppet samt vilka innebörder arkitekturen har för utvecklingsprocessen och hur dessa yttrar sig. De områden som varit av intresse introduceras tillsammans med problemområde och i relation till bakgrunden om varför de valdes i första hand. Uppsatsens disposition följer i övrigt den kvalitativa rapportens huvudlinje nämligen den linjära dispositionen med upplägget introduktion, problem (frågeställning), metod, teori, resultat och diskussion för att på så sätt borga för logisk och strukturell sammanhållning (Backman, 1998).
Begreppet systemarkitekturs innebörd och benämning varierar och begreppet belyses därför inledningsvis utifrån ett akademiskt perspektiv där dess grund tas upp. För att erhålla en snävare och därmed mer användbar definition av begreppet kommer det sedan att belysas ur ett mer funktionellt perspektiv. För det ändamålet har systemutvecklingsprocessen RUP valts eftersom den starkt fokuserar på just arkitekturell utformning och tillämpning, i processen är systemarkitekturen främst en produkt från arbetsflödet för analys och design med sin början i arbetsflödet för krav. Uppsatsen övergår till att beskriva begreppet systemarkitektur då redogörandet för RUP kommit till en punkt där det blir naturligt att övergå till systemarkitektursbegreppet.
Hur arbetet formats kan annorlunda uttryckas med hjälp av figur 1 nedan.



Figur 1. Modell över uppsatsens disposition.

1.6. Metod

Arbetet kan sägas tillhöra den hermeneutiska forskningsmetodiken då det inte passar in under den positivistiska skolan i någon betydande omfattning. Det har inte utgått från en grundläggande tanke om att arbetet skulle generera en absolut och objektiv kunskap bestående av hårda och atomära fakta (Wallén, 1996). Inte heller anses kunskap vara inhämtad enligt de två, för positivismen dominerande, källorna empiri och logik i någon större tidsmässig omfattning (Wallén, 1996). Med det menas att inga eller få empiriska iakttagelser eller studier av de valda ämnesområdena och företeelser som finns däri utförts. Snarare är det så att uppsatsarbetet bedrivits på ett hermeneutiskt sätt eftersom den mer är ett kvalitativt arbete framför ett kvantitativt då avsikten varit att på ett utredande sätt studera innebörder och ömsesidiga beroenden (Wallén, 1996) genom företrädesvis litteraturstudier. Men viktigt att betona är att betydande och starkt bidragande vägledning gällande studierna erhållits i intervjusamtal med metodexpert Sigvard Svensson samt sakkunskapskontroll av densamme och Mats Wessberg. Om än detta i uppsatsprocessen utgör empiriska inslag är huvuddelen av arbetet, åtminstone tidsmässigt sett, en kvalitativ litteraturstudie. Vidare har uppsatsuppgiften utförts på ett

subjektivt sätt med utgångspunkt från egna kunskaper och insikter vilka varit grundläggande för val av undersökningsområden samt fokus. Detta understryks vidare av åsikten om att det använda studiematerialet åtminstone delvis också måste ha utformats efter respektive författares subjektiva sinne och omdöme. I arbetet har det också eftersträvats att återge ömsesidiga innebörder och beroenden ur ett holistiskt perspektiv framför ett mer tekniskt av den anledningen att det ter sig mer intressant ur ett akademiskt perspektiv. Studierna av valda problemområden[3] kan i efterhand sägas ha karaktärsdrag utifrån tre olika typer av kvantitativa studier, nämligen: explorativa-, deskriptiva- och förklarande studier vilka alla främst relateras till undersökningar av icke kvantifierbara områden eller fenomen (Wallén, 1996). Ytterligare anledningar är också att ett av de studerade områden, systemarkitektur, på många olika sätt är ett mångfacetterat och därmed otydligt begrepp vilket beror på problemets natur.

1.6.1. Tillvägagångssätt och urval

Områdenas betydelser och innebörder för varandra har förtydligats successivt och därmed kan vald frågeställning sägas vara ett resultat av efterforskningar som visat sig bli vägledande för uppsatsens centrala delar samt övrig utformning. Detta snarare än att en i förväg framarbetad och därmed given frågeställning från början väglett det fortsatta utredningsarbetet med syfte att svara på frågeställningen. Anledningen ligger till stor del i den kvalitativa rapportens natur där det ofta är svårt att från början klart och tydligt definiera en vägledande fråge- eller problemställning eftersom det kan likställas med svårigheten att i förväg förutse en studies resultat utan föreliggande teorier. I övrigt är det också svårt att från start bilda en för arbetet normgivande teori utan någon större tidigare kunskap om de valda ämnena. Vidare har arbetet visat sig alternera mellan ett induktivt och deduktivt slutledningssätt beroende på aktuell situation. I de initiala skedena har främst den induktiva slutledningen varit dominerande just av den anledning att kunskap och insikt om områdena saknades, både var för sig men även om deras relation till varandra. Därför har arbetet inledningsvis skett utan föregående teorier eller hypoteser men vartefter inläsning skett och förståelse av ämnesområdena erhållits har även ett deduktivt slutledningsförfarande infunnit sig med sin grund i teoribildning utifrån hypoteser och orsakssamband (Thurén, 1995). Data har i ett relativt oavgränsat problemområde samlats in i syfte att, efter växande kunskap och insikt, inleda arbete med att kraftigt reducera dess omfång genom lämpliga avgränsningar samt val av önskvärt fokus i relevans med grunddragen av en möjlig frågeställning.
Frågeställningarna antogs ha flertalet möjliga svar vars konkretiseringsgrad kunde variera med vilken teknisk respektive akademisk nivå en fråga och dess svar låg på. Ju mer teknikinriktad nivå desto mer definitivt antogs svaret eller svaren på frågeställningen bli motsatt ett svar på en frågeställning på en mer abstrakt nivå.
Valet av studieunderlag har styrts utifrån ett tänkande där abstraktionsgraden av ämnesområdena varit avgörande för att på ett så rationellt sätt som möjligt införskaffa nödvändiga kunskaper för arbetets fortgång. Abstraktionsgraderna har successivt anpassats efter den aktuella situationens behov vilket initialt inneburit en relativt linjär litteraturstudie där författaren börjat med en allmän grundläggande orientering av

[3] Begreppen problemområde och undersökningsområde är synonyma med varandra.

ämnesområdena RUP och systemarkitekturer för att successivt övergå till något mer ingående och situationsanpassad litteratur. Grundläggande litteratur har studerats för att få insikt om hur dagens systemutvecklingsprocesser, primärt RUP, genom åren härletts ur det objektorienterade systemutvecklingsparadigmet samt hur begreppet systemarkitektur har sin normgivande historia inom systemparadigmen och de arkitekturella designteorierna. Studier har från början mest ägnats åt att undersöka grundläggande och generella ramverk för systemutvecklingen för att övergå i mer specifika ramverksstudier där Unified Process (UP) först sågs vara av intresse för att lägga en grund för ytterligare specificerande litteratur för RUP självt då den förra ligger som grund för den sistnämnda. Därefter har abstraktionsgraden i litteratursökandet anpassats i syfte att närmare, men fortfarande på ett grundläggande plan, erhålla förståelse och insikt om vad systemutvecklingsprocessen RUP är samt hur den är strukturerad. Efterhand som studier av processen visat hur centralt begreppet arkitektur är har inläsningen ändrat fokus från att vara generell, och omfatta hela processen, till att primärt undersöka sambanden dem emellan samt hur deras förhållande spelar in på frågeställningarna och deras svar. Parallellt med detta har litteratur, eller valda delar ur litteratur, med fokus på systemarkitekturer studerats för att få ett grundläggande grepp om vad en systemarkitektur är och vad den har för grund i de arkitekturella designteorierna. I övrigt kan litteraturen delas in i två huvudtyper; den studerande och undervisande litteraturen samt den mer redovisande litteraturen vilka kompletterat varandra relativt bra. Att viss litteratur i stort har samma ursprungskälla (läs RUP) har ibland gjort det svårt att variera angreppsvinkel på problemområdet samt redovisning av teori.

2. Rational Unified Process – en översikt

RUP står som förkortning till Rational Unified Process vilket är en heltäckande system och/eller programvaruutvecklingsprocess[4]. Processens olika grundläggande integrerade moment är inte unika för produkten RUP i sig utan återfinns i andra utvecklingsmetoder, processer eller ramverk som till exempel Checklands Soft Systems Methodology (SSM). Anledningen är att man under utvecklingen av RUP försökt att undvika att uppfinna hjulet på nytt och därför haft ambitionen att ur tidigare erfarenheter arbeta fram ett antal bästa praxis som grund för processens konstruktion (Lunell, 2003).
Företaget Rational har enligt en del endast vidareutvecklat konceptet till en kommersiellt gångbar produkt medan andra ser det mer som en unik produkt vars grund återfinns i UP. Vilken åsikt man än är av ändrar det inte att RUPs innebörd för ett systemutvecklingsprojekt fortfarande är heltäckande på en aggregerad- såväl som specifik detaljnivå samt ger utrymme för anpassning och justering till aktuella behov samt integration av projektstyrningsmetoder, exempelvis Ericssons Project Operation and Planning System eller Projektet för projektstyrning (PROPS) (Lunell, 2003). RUP självt följer inte någon specifik teknik för projektstyrning utan fokuserar på vad som är viktigt för processen vilket bland annat är fas- och iterationsplanering, riskhantering och resultatmätning (Lunell, 2003). Vad som avses med att RUP är heltäckande är att det inom produkten finns stöd att tillgå för, eller utrymme för stöd, i alla delar av utvecklingsprocessen. De stöd som finns antar varierande former men deras gemensamma och övergripande syfte är att definiera, förklara och fördela de arbetsuppgifter och medföljande ansvarsområden som förekommer i ett systemutvecklingsprojekt samt medel av olika slag för att kontrollera dem och dess kringrymd. Processen och alla dess ingående stöd grundar sig på normgivande principer men det finns också explicita utvecklingsverktyg för arbetet med specifika uppgifter. Vidare stödjer RUP användandet av Unified Modeling Language (UML) i exempelvis visuella modelleringsmoment.
En grundtanke bakom RUP är att den ska vara mycket dynamisk för att tillåta att ett gott arbete och resultat kan erhållas ur även mycket omfattande och komplexa situationer genom ett överlag iterativt och anpassbart tillvägagångssätt. Som exempel på detta kan nämnas att RUP i sin utformning tillåter användande organisation att göra ett val mellan två alternativa vägar; antingen att anpassa RUP efter omständigheterna eller anpassa omständigheterna efter RUP (RUP, 2001). Möjligheten att kombinera de två synsätten finns på olika nivåer i processen, strategisk, taktisk och/eller metodologisk men ju lägre nivå av hierarkin det gäller desto fler möjligheter finns det att variera mellan de två. Det görs fler metodanvändningsbeslut i exempelvis implementationsfrågor än det görs strategiska beslut angående projektets helhetsutformning (Lunell, 2003). Ett sätt att sluta sig till den uppfattningen kan vara att se till antalet inblandade intressenter på de olika nivåerna i ett utvecklingsprojekt.

[4] Begreppet systemutvecklingsprocess kommer vara synonymt med programvaruutvecklingsprocess om annat ej anges. Likaså förkortas de båda uttrycken med termen ’process’ där prefixen systemutvecklings- samt programvaruutvecklings- anses som redan införlivade i sammanhanget eller hämmande för textens läsbarhet.

Att RUP sägs kännetecknas av begreppen objektorientering, dynamik, flexibilitet och iteration möjliggörs på flera olika sätt beroende på vilken nivå samt del av processen man betraktar. Något eller några har redan nämnts kortfattat ovan men en för dynamiken viktig och nästan grundläggande faktor som hittills har utelämnats är valet av systemarkitektur[5] (Lunell, 2003). RUP fokuserar tidigt i utvecklingsprocessen på val av och tillämpning av systemarkitektur som en förutsättning för att skapa en dynamisk men robust, komponentbaserad och modulärt uppbyggd slutprodukt. Samtidigt är arkitekturvalet även en av förutsättningarna för att nå ett gott slutresultat på ett välstrukturerat sätt. Trots att RUP är en anpassningsbar process så är valet och användandet av systemarkitektur något som inte kan negligeras eftersom mycket av det som kännetecknar moderna objektorienterade beskrivningssätt infogas i RUP via en arkitektur vilket också utgör ett för processen grundläggande element (Lunell, 2003). Alternativt ser man det som att mycket av tankarna bakom och inom arkitekturbegreppet infogas med hjälp av objektorienterade beskrivningssätt.
Övrigt som är beaktningsvärt är att se RUP ur ett top-down perspektiv där man vid första anblick tycks kunna bryta ner dess ingående delar i evighet. Ett typisk drill-down förfarande där i hierarkin högre, mer strategiska och aggregerade nivåer ger upphov till en stor mängd olika abstraktionsnivåer som i sin tur möjliggör val av önskat betraktelseperspektiv som appellerar till olika intressentgrupper (RUP, 2001). Perspektiv som kan varieras efter betraktarens eller intressentens kognitiva förmåga, individens åtaganden samt uppgiftens krav.
Vidare beskrivning av RUP ger utrymme för att något mer ingående, men fortfarande grundläggande, förståelse för hur processen är sammansatt. RUP kan sägas bestå av två huvuddelar: en statisk elementstruktur som består av tillgängliga och i många fall rekommenderade processelement för ett arbetsflöde. Den statiska elementstrukturen krävs för att skapa den dynamiska processtrukturen, den dynamiska processen är en aktuell projekttillämpning av den statiska mallen (Svensson 2004-05-20). De statiska element som väljs för att utgöra en projekttillämpning består av för systemutvecklingsprocessen mer eller mindre karaktärsgivande drag. Där den statiska sidan som vars namn antyder står för de mer karaktärsgivande beståndsdelarna medan den dynamiska processtrukturen till större del består av anpassningsbara delar (RUP, 2001).

2.1. Grundläggande bästa praxis

Utvecklarna av RUP har inte haft för avsikt att återuppfinna hjulet på nytt utan har grundat konstruktionen av utvecklingsprocessen enligt gängse bästa praxis vilka man arbetat fram från tidigare utvecklingsprojekt och identifierat som viktiga framgångsfaktorer (Lunell, 2003). Inspirationen till dem är hämtade ur vitt skilda utvecklingsprojekt men vad de har gemensamt är att de och deras grunder är väl beprövade sedan tidigare och utgörs av principer, processer, metoder och modeller (Kruchten, 2002).

[5] Systemarkitektur kommer i sammanhanget ses vara synonymt med mjuk- och/eller programvaruarkitektur. För läsbarhetens skulle kommer termen ’arkitektur’ användas då specificering anses onödig.

De tillsammans med RUPs övriga nyckelegenskaper utgör stora och vitala delar av processen och utgör därmed mycket av grunden för dess statiska struktur. Det vill säga sådant som inte kan eller bör förändras i alltför avgörande mening. Dock kan valet tillämpningsanpassning göras efter aktuell systemutvecklingssituation i rådande projekt.

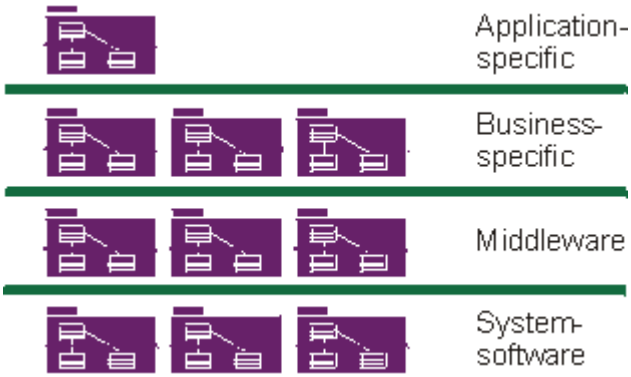
1. **Iterativ utveckling** är en av RUPs mest kritiska ståndpunkter vilken och man kan säga att en iteration består av ett eller flera minivattenfall (Wessberg, kommentar 2004-05-23). Inom ramen för iterativ utveckling (se Figur 2) finns också inkrementell eller stegrande utveckling som ett av flertalet olika sätt att ordna själva iterationerna - man skalar upp arbetet och resultatet efterhand. (Lunell, H., 2003) Det iterativa arbetssättet är vanligtvis överlägset den linjära eller vattenfallsmodellen, några av skälen är att det tillåter att man tar hänsyn till kravförändringar. En iterativ och inkrementell utveckling ger också en kontinuerlig integreringsprocess med verifiering vilket i sin tur leder till reducerad osäkerhet och problemrisk. (Kruchten, 2002)

Figur 2. Illustrerar tanken med iterativ utveckling (RUP, 2001).

2. **Kravhantering**.
   Ett av flera skäl bakom den iterativa utvecklingsmetoden är faktumet att krav hela tiden förändras från en tidpunkt till en annan och för att hantera detta erfordras kravhanteringsmetoder (Lunell, 2003), exempel på en arkitekturellt orienterad kravmodell är FURPS+ (Wessberg, kommentar 2004-05-23). Fokus på kravhanteringsmetoder leder till ett systematiskt sätt att få fram, organisera, kommunicera och hantera föränderliga krav på ett konstruktivt sätt. Det ger mer kontroll över komplexa utvecklingsprojekt samt förbättrad produkt- och processkvalitet. (Kruchten, 2002)

3. **Arkitekturcentrisk och komponentbaserad utveckling**
   Är en faktor som direkt påverkar om man lyckas med utvecklingsprojektet eller inte samt produktens kvalitet och livslängd. I de fall när programvaran utvecklas iterativt och inkrementellt är det än viktigare med en god arkitektur, den ska alltså därför vara en central fråga. (Lunell, 2003) Enligt gällande nyckelegenskaper drivs hela processen i stort av användningsfall men inom designaktiviteterna i

etableringsfasen (Elaboration) dominerar arkitekturbegreppet. Redan på ett tidigt stadium i processen är ett huvudmål inom iterationerna att skapa och utvärdera en arkitektur vilket tidigt kan resultera i en exekverbar arkitekturprototyp som gradvis ska utvecklas till att bär upp ett färdigt system under iterationernas gång. Inom RUP finns ett metodiskt och välorganiserat sätt att designa, utveckla och utvärdera en arkitektur. Det finns mallar, hjälpmedel, principer, verktyg, aktiviteter med mera för att ta fram en ur flertalet synvinklar hållbar systemarkitektur. Alla aspekter i framtagandet av en systemarkitektur finns redovisade inom processen. (Kruchten, 2002)
Komponentbegreppet och arkitekturbegreppet är beroende av varandra då en arkitektur byggs upp av komponenter och komponenters organisation utgör en arkitektur (Kruchten, 2002) se också figur 3. Att bygga en komponentbaserad arkitektur är en av RUPs grundläggande praxis men det är inte ett måste (Wessberg, kommentar 2004-05-23). En komponent är i huvudsak en fristående del av ett program eller ett system. En komponent ska vara så pass självständig, fristående och därmed utbytbar. Vidare ska den också ha en väldefinierad, tydlig och icke-trivial funktion i det program eller system den ingår i. Den ska även realisera ett väldefinierat gränssnitt enligt vedertagen specifikation och den kan vara exekverbar såväl som icke-exekverbar (Lunell, 2003). En implementerad abstraktion ur en design av något slag. Tillsammans med bildar den arkitektur- och komponentbaserade utvecklingen en modulär struktur där de ingående elementen kan designas, utvecklas och testas individuellt för att slutligen integreras i övriga systemet. Användningen av komponenter och systemarkitektur kan även möjliggöra storskalig återanvändning inom utvecklingsprocessen. (Kruchten, 2002)



Figur 3. Illustrerar komponentbaserad arkitektur i skikt. Mapparna är delsystem indelade i skikt, i mapparna visas komponenter (RUP, 2001).

4. **Visuell modellering**
   Sker exempelvis i UML och innebär att systemet kan betraktas från olika perspektiv beroende på vem individen är. UML tillåter att skilda aspekter av systemet synliggörs i ett och samma sammanhang (Lunell, 2003). Mycket av arbetet i RUP går ut på att utveckla och underhålla modeller av det system som ska utvecklas och dess kringmiljöer. Modeller underlättar förståelse,

problembeskrivning och lösning genom att erbjuda en förenklad bild av verkligheten och därmed också hantering av komplexitet. (Kruchten, 2002)

5. **Kontinuerlig verifiering**
   Om fördelarna med en iterativ utvecklingsprocess ska kunna tas till vara krävs en kontinuerligt planlagd verifiering som ingår iterationerna för verifiering av inkrementella leveranser. I annat fall är risken överhängande att små fel snabbt eskalerar och blir ohanterliga och då försvinner mycket av fördelarna med en iterativ och inkrementell utvecklingsmetod (Lunell, 2003). En iterativ utvecklingsmetod leder nästan automatiskt till kontinuerlig verifiering eftersom en iteration kan ses som ett miniprojekt med alla dess faser som exempelvis test (Kruchten, 2002).

6. **Konfigurations- och ändringshantering** är nödvändigt i en situation där iterativ och inkrementell utveckling råder eftersom produkten kommer att finnas i många olika utföranden med varierande funktionalitet och dylikt. Detta måste naturligtvis hanteras så att inget arbete råkar gå förlorat eller tid går till spillo i form av onödigt dubbelarbete. (Lunell, H., 2003)

2.2. Övriga nyckelegenskaper

Utöver de allmänt vedertagna praxis som nämnts ovan bygger RUP även på andra principer som är mer specifika för processen självt och därmed tillför mer distinkta egenskaper.

1. **Användningssfallsdriven process**
   Synsättet betyder att det är kring användningsfallen mycket av utvecklingsarbetet i RUP kretsar. De är faktiskt så viktiga för processen att det är de som driver hela utvecklingen framåt. (Lunell, 2003) Processen låter användningsfallen beskriva systemets beteende och överbrygger ett gap mellan explicita systemkrav och andra mer designorienterade artefakter. Men i de fall där det finns ett gap mellan användningsfallsmodellen (Use Case Model) och designmodellen är det möjligt att vidare överbrygga den med hjälp av användarfallsrealiseringar (Use Case Realizations) (Wessberg, kommentar 2004-05-23). I andra fall kan scenarios som är en kombination av ett eller flera användningsfallsflöden användas (Wessberg, kommentar 2004-05-23) eller liknande. (Kruchten, 2002)

2. **Riskfokusering**
   Är en annan viktig ståndpunkt som RUP håller på vilken innebär att man tidigt identifierar riskfaktorer som ouppmärksammade kan leda till problem och aktivt arbetar för att minimera dem. (Lunell, 2003) Risker är ofta kopplade till olika typer av krav funktionella såväl som icke-funktionella och de i sin tur ligger bakom risker med applikaitonsutvecklingen. Det finns många olika riskområden i hela utvecklingsprocessen varav arkitekturella risker är ett av dem vilken klassas som en så kallad teknisk risk. Andra risker är av mer administrativ art. Tekniska riskers grund kan härledas till olika användningsfall eller scenarier där

riskmomenten kan elimineras i en realisation (Wessberg, kommentar 2004-05-23).

3. **Processanpassning**
   Att kunna anpassa processen efter behoven kan göras på flera olika sätt eftersom RUP är konstruerat på så vis att delar antingen kan tas bort och ersättas eller anpassas. (Lunell, 2003) I annat fall om processen följs allt för strikt kan det leda till mycket onödigt arbete läggs ned på bitar man skulle klara sig bra utan. Som tillskott till processen kan man utnyttja den tillämpande organisationens policys, praxis, regler etc. (Kruchten, 2002)

4. **Verktygsstöd**
   Eftersom RUP gör anspråk på att vara en heltäckande systemutvecklingsprocess finns en mängd tillhörande verktyg som kan användas för specifika ändamål eller större övergripande bitar. De kan till exempel användas för att automatisera många steg i aktiviteterna. Processen erbjuder också verktygsguider för användning av verktygen. (Kruchten, 2002) Dock bör det noteras att RUP inte kräver att Rationals verktyg är just de som används utan processen tillåter att även andra verktyg används vilket i princip gör processen verktygsoberoende (Wessberg, kommentar 2004-05-23).

2.3. RUPs uppbyggnad – två dimensioner

- **Statisk struktur – vertikal dimension, processinnehåll**

RUPs statiska struktur står för de element vilka utgör delar i utvecklingsprocessens slutgiltiga sammansättning. Den definierar alltså hur processen är uppbyggd samt av vilka nödvändiga delar, den statiska strukturen kan sägas bestå av en fördefinierad verktygslåda vars element kan användas i en situationsspecifik projekttillämpning. Den statiska strukturens element förverkligar exempelvis de grundläggande bästa praxis och nyckelegenskaperna ovan. Man kan säga att disciplinerna representerar en realisation av det som de grundläggande bästa praxis och nyckelegenskaperna står för. RUPs statiska struktur är grundläggande för den dynamiska strukturen eftersom dynamiken är en projekttillämpning av valda delar och element ur den statiska strukturen.
Innehållet i RUPs vertikala dimension utgör förutsättningar för projektets utformning och utvecklingens dynamik då projekttillämpning är användningen av de statiska elementen efter aktuell situation. Innehållet i den dynamiska projekttillämpningen beror mycket på aktuell projektlivscykel. Man kan också beskriva den statiska- respektive dynamiska strukturen i ett innehålls respektive tidsperspektiv, content respektive time. (Svensson, samtal 2004-05-19)

- **Dynamisk struktur – horisontell dimension, tidsaspekten**

Utvecklingsprocessens dynamiska struktur har motsatt den statiska en mer följsam och adaptiv framtoning. Innebörden är att om det är den statiska strukturen som ger förutsättningar för utvecklingsprocessens dynamik så är det via den dynamiska strukturen processens anpassbarhet förverkligas. Det är främst genom den, när den statiska strukturen redan är beslutad om, som anpassning av utvecklingsprocessen efter aktuella

och föränderliga förhållanden görs. Den dynamiska strukturen är uppbyggd kring fyra olika faser som tillsammans utgör en utvecklingscykel vilken i slutändan producerar ett resultat av något slag.
Den dynamiska strukturen består av faser vilka beskrivs översiktligt utifrån RUP (2001) och är inte komplett återgivna eftersom det inte är nödvändigt för en grundläggande förståelse av processen och dess uppbyggnad tillsammans med iterationer som är utgör situationsanpassad projekttillämpning av de statiska processelementen. Faserna som återges nedan är grundläggande för RUPs iterativa och inkrementella arbetssätt.

1. **Inception phase** (Förbereldelsefasen)
   Mål:
   Ena alla intressenter om projektets livscykelmål. Här specificeras saker som affärsmål, visioner, projektomfattning vilka visas som livscykelmål. Definiera projectscope som anger projektets omfattning och ramverk. Urskilj systemkritiska användningsfall, det vill säga primära scenarios som kommer driva systemet och ligga till grund för viktiga designkompromisser (RUP, 2001). Utkast till åtminstone en kandidatarkitektur som relaterar till de primära scenarierna.

   Viktigaste aktiviteter rörande arkitekturen:
   Prioritera användningsfall (Prioritize Use Cases) vilket innebär en prioritering av de användningsfall som kommer att inverka på och driva arkitekturens framkomst, det vill säga att man tar fram de arkitekturellt signifikanta användningsfallen (Wessberg, kommentar 2004-05-23).
   Sammanställ ett arkitekturförslag med målet att demonstrera projektets utförbarhet genom ett slags konceptbevis (Architectural Proof Of Concept). I förslaget ska även ingå förslag till möjliga designkompromisser samt komponenttänkande i form av vad som kan tänkas gå att köpa in eller återanvända.

   Milstolpe: livscykelmål.
   För att utröna projektets livsduglighet.

2. **Elaboration phase** (Etableringsfas)
   Mål:
   Analysera problemområdet, bygg en arkitekturell grund, utveckla projektplan och genom detta uppmärksamma och reducera projektets största riskmoment. Den mest kritiska av de fyra faserna, leder till beslut om projektet ska föras vidare eller inte. Den arkitekturella grunden ska utgöra en stabil bas för majoriteten av designen och den efterföljande implementationen i konstruktionsfasen (Implementation). Arkitekturen växer fram ur de viktigaste systemkraven exempelvis arkitekturellt signifikanta användningsfall (Use Case) tillsammans med riskövervägande. Säkerställ att arkitekturen, kraven och planerna är stabila samt att riskerna minimeras nog att bestämma kostnader och tidsramar för projektet. Adressera de betydande riskerna man kartlagt för projektet.
   Demonstrera att den grundläggande arkitekturen man tagit fram kommer att klara av att stödja och förverkliga systemet inom givna tids och budgetramar. Arkitekten är

nästan projektledare i den här fasen eller arbetar mycket tillsammans med den egentliga projektledaren eftersom arkitekturen är av en så pass central betydelse i den här fasen, man arbetar mycket med arkitekturell prioritering respektive funktionell prioritering. (Svensson, samtal 2004-05-19).

Viktigaste aktiviteten rörande arkitekturen:
Definiera och utvärdera en kandidatarkitektur (Candidate Architecture) så fort som möjligt.
Förfina arkitekturen (Refine Architecture) och välj ut komponenter. Komponenter som ses som möjliga alternativ utvärderas tillräckligt väl för att beslut om köp, återanvändning eller utveckling ska kunna göras i tillräckligt stor utsträckning för att ytterligare kunna definiera tidsramar och kostnader.
De valda komponenterna integreras och utvärderas gentemot de primära scenarierna.

Milstolpe: livscykelarkitektur.
Man vill ha en etablerad och stabil version av arkitekturen, inte en grundversion på så vis att den ska behöva vidareutvecklas, utan en stabil arkitektur så att det inte blir några ändringar. Arkitekturen ska vara färdig för applikationens arkitekturella krav, det ska inte tillkomma något arkitekturellt konstruktionsarbete som man i förväg känner till. (Svensson, samtal 2004-05-19)

3. **Construction phase** (Konstruktionsfas)
Mål:
En tillverkningsfas där fokus ligger på styrning och kontroll av resurser, arbete, processer med mera för att optimera kostnad, tid och kvalitet i realiseringen av systemet enligt tidigare analys och designresultat. Hela konstruktionsarbetet stödjer sig på den grundläggande arkitekturversionen som man tidigare arbetat fram under Elaborationsfasen. Arbetet övergår från att kretsa kring analys och design samt utveckling av intellektuella abstraktioner till att ta form som konkreta tillämpningar. Vartefter de återstående komponenterna av systemet blir klara utvärderas de, testas och integreras i den framväxande produkten.
Beroende på projektets storlek kan man uppnå en viss grad av parallellism i implementationsarbetet vilket tillför komplexitet och ställer krav på den arkitektur man använder sig av. Arkitekturen måste till exempel tillåta modularitet och komponentbaserad utveckling för att parallellism ska fungera i realiseringsarbetet.

Viktigaste aktivitet rörande arkitekturen:
Ingen specifik.

Milstolpe: initialt funktionsduglig version av produkten.
För att avgöra om programvaran, driftsmiljön och användarna är tillräckligt redo för driftsättning utan att riskera projektet. Kortare uttryckt blir resultatet en betautgåva.

4. **Transition phase** (Överlämninsfas)
   Mål:
   Att överlämna en programvaruprodukt till användarna. Fasen kan sträcka sig över flera iterationer och anses avslutad när produkten uppfyller projektets krav och en vision av den har levererats. Fasen är tänkt att behandla sådant som blivit över eller sparat under projektets gång för att tas om hand i projektets slutskede. Vad som behandlas beror på hur väl produkten överensstämmer med ställda krav och förväntningar men tanken är inte att det som behöver åtgärdas ska vara av alltför kritisk art, snarare är det fråga om justeringar. Fasen kan vara enkel eller komplex vilket återigen beror på vad dess mål är att få utfört, det kan röra sig om lösningar på mindre problem eller att helt färdigställa egenskaper som tidigare inte varit helt kompletta till att addera nytillkommen funktionalitet.

   Viktigaste aktivitet rörande arkitekturen:
   Ingen specifik.

   Milstolpe: produktutgåva.
   Vilken man verifierar gentemot ställda krav på produkten för att avgöra huruvida projektet lyckats med sina mål eller inte.

- **Om faserna generellt**

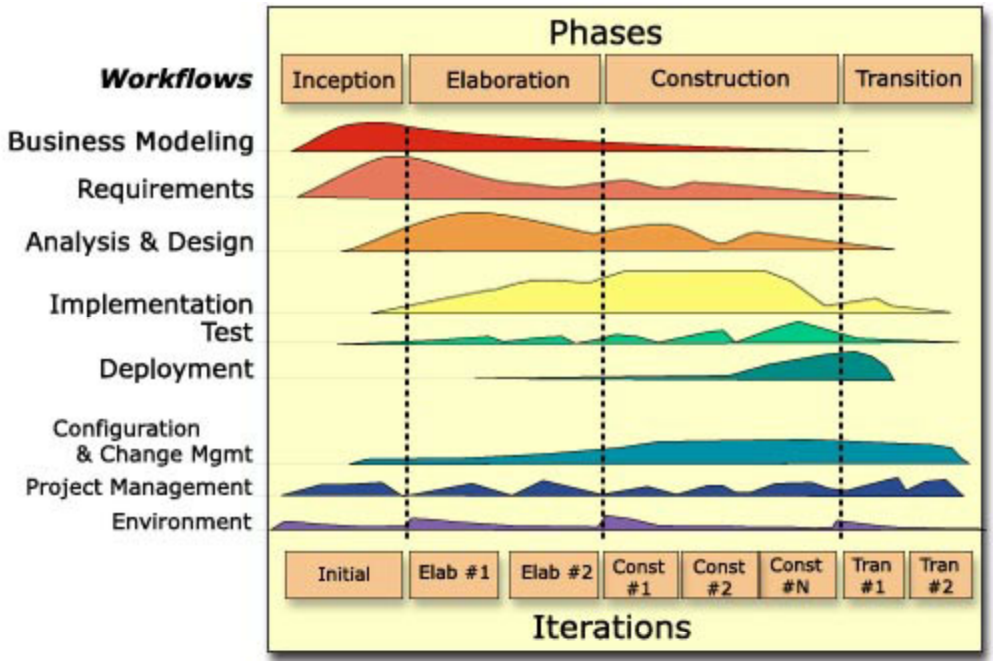
Faserna är olika långa och innebär även olika arbetsbelastning och intensitet. Vidare har de också olika förutbestämda mål och syften vilka ger dem deras varierande karaktärsdrag så som deras utformning och de milstolpar deras utförande skall uppfylla. Som sagt utgör de tillsammans en utvecklingscykel och innehåller delar av en eller flera iterationer. (Lunell, 2003) Den fas man befinner sig i kan sägas definiera projektets temporala och reella status och även visa på eventuella brister i projektet, bedriver man arkitekturellt utvecklingsarbete i konstruktionsfasen borde utvecklingen nog befinna sig i en tidigare fas (Wessberg, kommentar 2004-05-23).

- **Om utvecklingscyklerna generellt**

Ofta finns det flera utvecklingscykler inom samma projekt då de syftar till att utveckla det som så småningom ska färdigställas på varierande nivåer. Utvecklingsarbetet påbörjas i en initial utvecklingscykel vars arbete leder till färdigställandet av en produktversion vilken fortsättningsvis vidare färdigställs i takt med att följande vidareutvecklingscykler påbörjas och avslutas. Cyklerna kan i viss mån överlappa varandra men då endast i varandras överlämnings- respektive förberedelsefas. (Kruchten, 2002) Det är viktigt att påpeka hur utvecklingscyklerna skiljer sig i ett projekt vars syfte är att skapa ett system från grunden jämfört med hur det ser ut i ett projekt där det är frågan om förvaltning eller vidareutveckling av ett redan befintligt system (Wessberg, kommentar 2004-05-23).

- **Om det iterativa arbetssättet generellt**

Se det iterativa arbetssättet och de olika utvecklingscyklerna som trappsteg mot ett slutgiltigt mål, en produkt att överlämna i överlämningsfasen. Det iterativa arbetssättet är inte något unikt för RUP men däremot ett gott exempel på en grundläggande bästa praxis.



Figur 4. Modell på RUPs övergripande tvådimensionella struktur (RUP, 2001).

De två dimensionerna, en vertikal och en horisontell, syftar båda till att uppfylla bland annat de sex bästa praxis som är så viktiga för processen.
Den vertikala dimensionen ”Workflows”[6] beskriver de nio olika huvuddiscipliner som finns i processen ordnade efter arbetsuppgifter medan den horisontella visar på hur de olika faserna ligger i tiden samt hur de består av en eller flera iterationer (RUP, 2001) se figur 4. Det är viktigt att notera att beskrivningarna av de olika disciplinernas arbetsintensitet och aktiva tider i processens faser och iterationer endast är ett typexempel på hur det kan se ut, det behöver alltså inte nödvändigtvis vara på just detta sätt som illustreras i figuren. Det finns skillnader i ett projekt som bygger upp ett system från grunden, där det inte finns något tidigare system att utgå från, och ett projekt som ska vidareutveckla ett redan befintligt system där till exempel arkitekturen finns sedan tidigare. Antalet iterationer är med andra ord helt och hållet beroende av hur det aktuella projektet ser ut (Lunell, 2003).
De sex första disciplinerna är så kallad tekniska arbetsflöden som alla är direkt involverade med konstruktion av en produkt, av den anledning kan de också kallas producerande discipliner, medan de tre sista är stödjande discipliner vilka mer syftar till att ordna och kontrollera de andra arbetsflödena i utvecklingsprojektet.

[6] Workflows, huvuddiscipliner, discipliner och arbetsflöden ses vara synonyma med varandra.

På den här nivån kan man tydligt se att ett visst släktskap råder mellan den iterativa systemutvecklingsprocessen RUP och andra vattenfallsmodeller. Trots allt är det inte släktskapet som är av störst vikt utan de mer betydande skillnaderna dem emellan. (Lunnel, 2003)

3. Systemarkitektur – systemets organisation

En systemarkitektur är den fundamentala organisationen av ett system som helhet. Aspekterna som tillsammans utgör arkitekturen inkluderar statiska såväl som dynamiska element, hur de samverkar sinsemellan inom systemet samt hur de utgör systemets beteende utifrån sett. Vidare inkluderas också systemets arkitekturella stil som visar de bakomliggande tankegångarna som lett fram till arkitekturens sammansättning. Arkitekturbegreppet involverar också andra egenskaper gällande systemets prestanda, skalbarhet, återanvändningsgrad, ekonomiska- och tekniska begränsningar.
(Scott, 2002)
Ursprungsbehovet av att använda sig av systemarkitekturer har vuxit fram med tiden allteftersom informationssystemen blivit allt mer komplexa vilket lett till allehanda svårigheter (Shaw, Garlan, 1996).

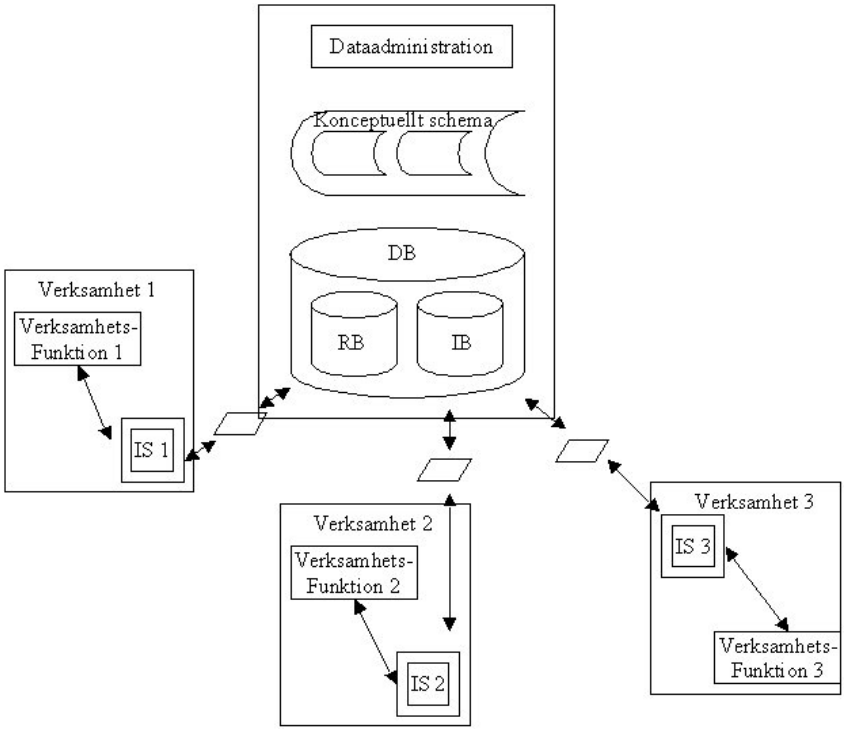
3.1. Bakomliggande designteorier för systemarkitekturer

För begreppet systemarkitektur finns ett antal bakomliggande designteorier vilka kan ses som grundläggande för systemarkitekturens struktur och utformning. Nedan kommer två dominerande och motstående teorier behandlas för att ge inblick i hur en systemarkitektur kan ha grundläggande influenser långt ifrån själva systemutvecklingsprocessens primära fokus och dess aktiviteter. Vad som slutligen resulterar i en systemarkitektur kan alltså ha sin upprinnelse i organisationers strukturella-, sociokulturella-, infologiska- och funktionella arkitekturer som i sin tur direkt härleds från organisationer som fenomen (Magoulas, samtal 2004-04-?). Eftersom de två teorierna är varandras motsatser är de fundamentalt åtskiljda i hur de ser på en given informationssituation samt kringliggande miljöer, oavsett hur begränsad eller omfattande den än är. Gemensamt är dock att de båda teorierna, om än olika, avser att genomgående förbättra organisationers informationsförsörjning då de betraktar informationen som en alltmer kritisk resurs som bör behandlas därefter (Magoulas, Pessi, 1998).
De systemarkitekturer som man ideligen använder sig av idag kan i olika stor utsträckning härledas från de två designteorierna; informationsbaserad – och verksamhetsbaserad designteori. Alternativt härleds designteorier från en given situation men oavsett från vilket håll man kommer är det klart att ju bättre klassificerad en arkitektur eller situation är desto lättare blir det att analysera den, designa och konstruera en lösning för den som är genomgående konsekvent enligt vedertagna regler. Det i sin tur ger möjlighet att reducera ett systemutvecklingsprojekts inneboende komplexitet överlag.

3.2. Informationsbaserad designteori

Den bakomliggande filosofin är här att man betraktar själva informationen som en central resurs för organisationen och därmed likställer den med andra mer traditionellt kapitala företagsresurser som maskinparker eller dylikt. Därför betonar teorin först och främst vikten av att man friställer informationen från andra områden vilka man ser som föränderliga (Magoulas, Pessi, 1998). Informationsbasen ska alltså frigöras från dess allehanda behandlingsfunktioner inom IS/IT, försörjande verksamheter samt sociala sammanhang (Magoulas, Pessi, 1998). Informationsresursen ska med andra ord

stabiliseras då man anser att den till sin natur är objektiv och därmed finit medan dess användningsområde kan variera. Så mycket som möjligt av informationsresursen ska vara globalt tillgänglig via en central administrativ och distribuerande funktion, den anskaffas en gång av den funktion som framställer den och på så vis anses resursen få en hög integritet (Magoulas, Pessi, 1998). Endast ett minimum av information bör lagras lokalt och det är främst operativ och starkt verksamhetsbetonad information som utgör en logisk och integrerad del av den lokala verksamheten som inte hör hemma centralt. Ett principiellt exempel på hur en IB-arkitektur kan se ut visas nedan i figur 5.

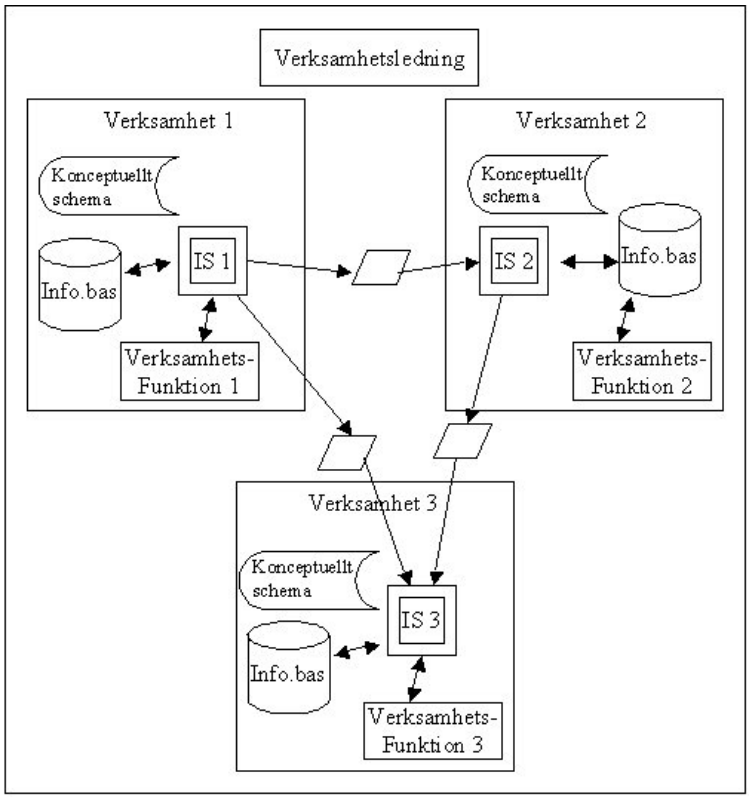


Figur 5. Typisk övergripande IBA, (Magoulas, Pessi, 1998)

3.3. Verksamhetsbaserad designteori

Teorin bakom den här typen av arkitekturer är strikt verksamhetsbaserad vilket innebär att organisationens olika verksamhetsdelar ska basera alla sina informationsbehov på egna relativt autonoma men samverkande informationssystem med lokala objektsystem, regel- och informationsbaser (Magoulas, Pessi, 1998). Detta leder totalt sett till ett flexibelt system där lokala uppdateringar och förändringar i systemet går smidigt eftersom det inte i någon större utsträckning ska ge upphov till följdeffekter och påverka helheten på ett omvälvande sätt. Varje verksamhetsdel har ansvar för sitt system och för informationsförsörjningen till detta samt den information andra, inklusive det centrala samordnade systemet, behöver från det egna lokala systemet (Magoulas, Pessi, 1998). Det finns alltså två distinkta typer av information; den lokala respektive den globala sambandsinformationen vilken flödar mellan organisationens olika verksamhetsbaserade informationssystem. Sambandsinformationen ska inte lagras eller göras globalt tillgänglig

utan enbart berörda parter ska omfattas av dess flöde. Utbytet av sambandsinformation sker genom en så kallad meddelandeprocess. Man strävar efter att koppla informationssystemen närmare ansvarsmässigt och logiskt avgränsade verksamhetsfunktioner och decentralisera de olika delsystemen för att bättre återge verkligheten eller objektsystemet (Magoulas, Pessi, 1998). Trots den höga graden av decentralisering är det mycket viktigt att också fokusera på samordning mellan de olika delsystemen annars kan det hela leda till kaos och ansvaret för en organisations IS-struktur som helhet ligger hos den centrala ledningen. Ett principiellt exempel på hur en VB-arkitektur kan se ut visas nedan i figur 6.



Figur 6. Typisk VBA, (Magoulas, Pessi, 1998)

4. Arkitekturens substans och betydelse för RUP

I den här delen av uppsatsen kommer arkitekturbegreppet att behandlas på ett mer tillämpningsbetonat sätt än i de bakgrundsgivande designteorierna ovan. Begreppet belyses ur ett mer konkret perspektiv inom systemutvecklingsprocessen RUP för att erhålla en mer pragmatiskt bild av begreppets inverkan. Vid den här punkten i uppsatsen är det författarens avsikt att närmare koppla samman arkitekturbegreppet med RUP och därmed grunda för besvarandet av respektive relaterade frågställningar. Återkoppling till designteorierna kommer ske där så är befogat.

4.1. En vag begreppsdefinition

Systemarkitektur är något som gör sig gällande på flera plan i ett informationssystem eller en mjukvaruprodukt oavsett storlek, komplexitet eller omfattning (Shaw, Garlan, 1996). Men i och med att begreppet används flitigt på flera olika plan, av skilda individer besittande olika roller och i vitt varierande utvecklingsprojekt får det naturligtvis konsekvensen att begreppets innebörd blir diversifierad (Kruchten, 2002). Förhoppningsvis är dock begreppet så tydligt definierat och därmed entydigt nog bland berörda parter i en viss situation att dess innebörd för projektet inte går förlorad till den grad att dess användande leder till negativa konflikt- och motsatsbetonade konsekvenser. Att begreppet inom systemvetenskapen har fått så bred prägling kan delvis ha sin förklaring i att dess användningsområden i sig inte är helt klart definierade och att deras inbördes samverkan, struktur och innebörd inte heller är klargjord i tillräcklig utsträckning (Kruchten, 2002). Ytterligare en bidragande orsak kan vara att uppfyllande av förutbestämda definitioner innebär ett visst mått av anpassning, en typ av åtagande och ansvar som kanske inte alltid är välkommet. Det kan i ett kortare perspektiv till synes innebära en omväg till målet men i ett längre och mer strategiskt perspektiv kan det istället innebära takiska nackdelar för utvecklingsprocessen och dess resultat. En annan grundläggande faktor som också kan ha bidragit till begreppets otydliga och inkonsekventa betydelse är att det en gång i tiden togs från byggsektorn och därefter anpassades som det sågs lämpligt istället för att ett för systemvetenskapen unikt begrepp arbetades fram. En för begreppet redan tidigare etablerad innebörd följde alltså med från början. (Scott, 2002) Peter Eeles är av uppfattningen att en annan svårighet med arkitekturbegreppet och dess tillämpning är att samla ihop och definiera grundläggande krav på den.

Vilken definition systemarkitektur än getts i en viss situation är det klart att begreppet är av stor vikt, vare sig det används eller inte, och därför bör ägnas åtanke i gängse proportion till dess innebörd för den aktuella situationen. Detta styrks också av det faktum att begreppet återfinns och har implikationer på så många olika platser och nivåer inom ett systemutvecklingsprojekt. Just den frekventa förekomsten, främst i första hälften, under analys och design, i en utvecklingsprocess, torde bekräfta även dess strategiska och taktiska betydelse i praktiken (RUP, 2001). Avslutningsvis en mycket viktig anledning till att det kan vara svårt att exakt definiera och beskriva vad en systemarkitektur är beror på att det inte är helt lätt att separera begreppet från analys och

designförfarandet eftersom det däremellan råder ett ömsesidigt beroendeförhållande (RUP, 2001).

4.2. Arkitekturbegreppet förtydligat i RUP

Arkitektur och design ska inte förväxlas med varandra då arkitektur är en del av design men all design är inte arkitektur. Arkitektur handlar om den högsta designnivån för ett system och dess omgivning.(Lunell, 2003) Inom RUP finns lite olika perspektiv på arkitekturbegreppet men totalt sett ges det en klar definition och funktion baserat på en holistisk systemsyn. Perspektiven beror på abstraktionsgraden i utvecklingsprocessen och varierar med betraktarens plats och funktion i utvecklingsarbetet. RUP avviker inte från definitionen vilket gjort att den är ganska bred och innehållsrik (Kruchten, 2002). I övrigt är det endast en eller en liten grupp som aktivt arbetar med att ta fram en lämplig systemarkitektur vilket minskar risken för att termen ges en otydlig och otillräcklig definition samt innebörd (RUP, 2001). Man undviker alltså problem som en mindre kontrollerad definition och användning av begreppet och dess innebörd medför. En situation där den diversifierade bilden av begreppet exempelvis bidrar till eller rent av verkar för kommunikationsproblem, onödiga analys och designproblem som får vittgående konsekvenser undviks på så sätt. Är också begreppets olika bilder, eller arkitekturvyer väl koordinerade, som är tanken, ska sagda problemsituation inte uppstå, åtminstone inte med ett oklart arkitekturbegrepp som bidragande orsak.

4.3. Arkitekturens huvudsakliga syften

Man konstatera att det är svårt att klart och tydligt definiera arkitekturbegreppet och på ett annat sätt försöka redogöra för dess innehåll och betydelser. Arkitekturen syftar till att beskriva och strukturera ett system i dess helhet. Uttrycket omfattar både statiska och dynamiska strukturer samt hur de och dess olika komponenter samverkar tillsammans samt den arkitekturella stil som återspeglar den användande organisationen. Arkitekturen berör också områden som skalbarhet, återanvändning, prestanda, ekonomiska och teknologiska begränsningar och möjligheter (Kruchten, 2002).

RUP beskriver begreppet som den fundamentala grundstruktur kring vilken systemet byggs och att det därför måste ses som att det är av central betydelse att projektet styr arbetet (främst inom design), dess upplägg och resultat, med arkitekturen som en central utgångspunkt. Ordagrant står i RUPs dokumentation ”the architecture of a software system (at a given point) is the organization or structure of the system's significant components interacting through interfaces, with components composed of successively smaller components and interfaces”. Av den anledningen är det av stor vikt att ansvariga för projektets iterationer låter realisationsarbetet kretsa kring den framtagna arkitekturen. Arkitekturen kan bland annat ses återspegla en vision av vad som ska konstrueras vilken måste delas och förstås av projektets samtliga medlemmar (Scott, 2002). Skulle man inte ha någon arkitektur eller en bristfällig sådan kan det resultera i att projektet misslyckas på flertalet punkter, inklusive sina mål, men mer konkreta effekter är problem med kommunikation, synkronisering inom och utom projektet, informationsåtkomst, skalbarhet och prestanda (RUP, 2001). De positiva effekterna som räknas upp nedan

riskerar att mer eller mindre gå förlorade och ersättas av negativa motsatser i avsaknad av en väl genomarbetad systemarkitektur.

- **Helhetsförståelse**

Arkitekturbeskrivningen[7] syftar i första hand till att främja förståelse av den arkitektur vilken systemet eller mjukvaran är uppbyggd kring så att samtliga berörda parter kan erhålla en helhetsbild av projektet och dess syfte. Detta jämfört med en situation där endast fåtalet projektmedlemmar är väl införstådda med systemets helhet. (RUP, 2001)

- **Organisera utvecklingsarbetet**

Arkitekturen beskriver explicit olika bitar av systemet samt dess ingående delar och deras inbördes relationer och gränssnitt mot varandra. Den innehåller även en eller flera arkitekturella mönster exempelvis klient/server, three-tier och n-tier, vilka bidrar till att strukturera och organisera utvecklingsarbetet. Genom den typen av arkitekturanvändning kan projektmedlemmarna försäkra sig om en bättre internkommunikation vilket i sin tur bidrar till höjd effektivitet. (RUP, 2001)

- **Möjliggöra återanvändning**

En aspekt av komponentbaserad systemutveckling är att komponenterna ska konstrueras på ett så standardiserat och generellt sätt som möjligt så att de kan återanvändas i varierande situationer med ett minimum av specialisering. Detta görs möjligt av en stabil och väl genomarbetad arkitektur i vilken komponenterna kan sättas samman och integreras med varandra efter behov. Arkitekturen ska även bidra till att sådana tillfällen och återanvändningsmöjligheter upptäcks och gynnas. Avsikten är bland annat att ju mindre tid utvecklarna behöver ta i anspråk för att utveckla nya komponenter desto mer tid finns till övers för att förstå kundens problemsituation och konstruera lösningar till den. (Kruchten, 2002)

- **Förutsättning för att vidareutveckla systemet**

Är ett viktigt steg i produktlivscykeln som väger tungt inom en systemarkitekts arbetsområde och bidrar i stor utsträckning till produktens användbarhet, integritet och aktualitet eftersom dess omvärld ständigt förändras och därmed också ändrar produktens existentiella villkor. Arkitekturen ska ur det här perspektivet tillåta att ändringar i en del av systemet inte automatiskt medför förändringar i en annan del så att man slipper komplicerande följdeffekter att ta hänsyn till. (Kruchten, 2002)

- **Hjälper att sålla ut relevanta användningsfall**[8]

Man skulle kunna säga att användningsfallen driver utformning och användningen av arkitekturen eller att tillämpningen av arkitektur direkt medverkar till att arkitekturellt signifikanta användningsfall väljs ut som motiverande grund för arkitekturens utformning (Wessberg, kommentar 2004-05-23). Användningsfallen definierar då inte enbart

[7] Se rubriken Artefakt samt artifakten Software Architecture Document.
[8] Termen användningsfall kommer ses vara synonymt med användarfall och engelska termen use case(s).

”klassisk” systemfunktionalitet utan används på så sätt även till att grundlägga en arkitektur vilken i sin tur grundlägger för en god ”klassisk” systemfunktionalitet. Fokus läggs alltså på de användningsfall som mest kommer bidra till att fylla ut arkitekturen i takt med att systemet realiseras och växer.
(Scott, 2002)

- **Ritningar ur olika perspektiv av systemet**

Ett av de främsta syftena som ligger bakom utformningen och användandet arkitektur är att få en överblicksbild som utgörs av flera olika ritningar. Detta för att se arkitekturen ur olika perspektiv, av systemet och därmed underlätta förståelse av detsamma samt att visualisera ritningar över det. Överblickbarheten i ritningarna gör det möjligt att på ett relativt enkelt sätt tillägna sig systemets uppbyggnad och bidrar därmed även med förståelse för när, var och hur en ny funktion, komponent eller element ska integreras i det tidigare systemet samt vilka följdeffekter det får.
Den så kallade ritningen visar alltså hur systemet är konstruerat, av vilka funktioner, mekanismer och komponenter, hur de samverkar sinsemellan och utåt. Utifrån detta kan man även sluta sig till systemets beteende i stora drag. Märk väl att en arkitekturbeskrivning är obeständig och i samma stund det aktuella systemet eventuellt förändras görs dess arkitektur icke-gällande. (Lunell, 2003)

- **Systemets samverkansprinciper**

Eftersom det ingår i definitionen av arkitekturbegreppet att visa hur ett systems olika delar samverkar sinsemellan såväl som mot externa företeelser måste den därför också fastställa principerna för samverkan och kommunikation. Det kan göras via överenskomna konventioner och protokoll för hur gränssnitt formellt ska utformas. (Lunell, 2003) Exempel kan vara RFC-protokoll (Request For Comment) för hur kommunikation över internet ska eller bör ske eller fjärranrop av funktioner RPC (Remote Call Procedure) eller ISOs OSI-modell för standardiserad datakommunikation (Clements, 2000).
Under den här punkten kan också nämnas hur dagens systemsamverkansprinciper anknyter till de grundläggande designteorier VBA och IBA som tagits upp tidigare i uppsatsen. Teorierna bakom VBA och IBA är segregerande och uteslutande till sin natur då den ena i stort sett utesluter användningen av den andra men i dagens läge är det synsättet inte lika aktuellt då man i mycket stor utsträckning fokuserar på integration. Man kan idag istället tillämpa de olika designteorierna i något annorlunda situationer där de båda används i samma systemdesign som komplement till varandra. VBA kan idag sägas stå för analys av de dynamiska verksamhetsdelarna där man kartlägger vilken information som hör hemma och används var, exempelvis hör användningen av användningsfall hemma här. IBA å andra sidan är mer implementationsinriktad då man med hjälp av dess teorier och principer kartlägger hur realiserande komponenter ska se ut. Att de båda designteorierna idag kan tillämpas samtidigt är att (verksamhets)komponenter numer har både logik- och dataansvar. Man frågar sig *vad* i verksamhetsrelaterade termer enligt VBA och *hur* enligt IBA i ett skikt- och komponenttänkande. (Svensson, samtal 2004-05-19)

- **Systemets utbyggnadsprinciper**

Ett system som byggts är inte helt aktuellt längre tid än dess omvärld är oföränderlig vilket innebär att det, som redan sagts, krävs en arkitektur som tillåter framtida förändringar och utbyggnad av systemet. Utbyggnadsprinciperna kan även dem vara av olika utformning, de kan vara konventioner som definierar hur nya komponenter ska införlivas i systemet och dess arkitektur. Det kan till exempel röra sig om hur ett nytt användarfall ska implementeras (ny systemfunktionalitet). Principerna för utbyggnad kan också grunda sig på mekanismer som förenklar förändringen av ett system, det måste då finnas en underliggande mjukvara som har till uppgift att underlätta ett förnyande av systemet. Även om de ses som skilda torde en mekanism i grund stödja sig på konventioner om hur utbyggnad ska ske. (Lunell, 2003)

- **Systemets arkitekturella mönster**[1]

Ett arkitekturellt mönster beskriver ett välbeprövat, framgångsrikt och dokumenterat tillvägagångssätt att etablera en god lösning på en problemsituation. För att det ska vara ett mönster krävs dock förutom redan nämnda saker också att det är ett tillräckligt generellt tillvägagångssätt för att kunna appliceras på återkommande liknande, men inte nödvändigtvis identiska, problem. Med andra ord handlar det om abstrakta och generellt tillämpbara lösningar på snarlika, men dock varierande, problem. Mönster kan användas på många olika nivåer av detaljupplösning men här tänker man främst i strategiska och taktiska termer ur ett helhetsperspektiv. Exempel på arkitekturella mönster är skiktad arkitektur där man skiljer på gränssnitt, data, logik och teknik och det finns även den traditionella klient-server-arkitekturen. (Lunell, 2003)

- **Systemets arkitekturella stil**

Beroende på hur man i en situation väljer att blanda de tidigare beskrivna delarna vilka tillsammans utgör, om inte hela så åtminstone, en ansenlig del av arkitekturbegreppet kan man sägas ge en arkitektur en viss stil. Stilen väljs naturligtvis utifrån olika utgångspunkter beroende på aktuellt projekt men försöker man att sätta samman arkitekturer efter samma princip man väljer mönster gagnar man en konsistent hållning. (Lunell, 2003)

Trots många definitioner av begreppets betydelse och innebörd verkar ovan definitioner enligt studerade källor vara allmänt vedertagna inom RUP, UP och UML.
Nedan följer i korthet ytterligare förklaringar och definitioner av vad en systemarkitektur är.

- Hantera systemets logiska organisation.
- Dess logik och funktionalitet.
- Hantera dess temporala aspekter, till exempel tidsberoenden mellan processer.
- Klargöra dess fysiska struktur samt hur den logiska strukturen (mjukvarusystemet) utnyttjar det.

[1] Mönster kan användas för att beskriva mekanismer

- Rationell kontroll för hantering av ett projekts komplexitet samt upprätthålla dess integritet.
- Utgöra en grund för projektledning. Planeringen och bemanningen organiseras till stor del efter huvudkomponenterna: skikt och delsystem.

4.4. Arkitekten

Rollen som arkitekt är aktuell på flera ställen genom hela utvecklingsprocessen men fokus på vad som återges här regleras av de avgränsningar som tidigare angetts för uppsatsen som helhet. Återgivningen av arkitektens roll sker på ett generellt plan för att därefter påvisas i de delar av utvecklingsprocessen som främst berör det arkitekturella grundarbetet och den arkitekturella konstruktionen.
Den viktigaste nyckelpersonen i framtagandet av arkitekturen innehar just rollen Arkitekt. Arkitektens ansvarsområde inom analys och design är kortfattat att leda och koordinera tekniska aktiviteter och artefakter i ett utvecklingsprojekt. Att slå fast en övergripande struktur för varje arkitekturell vy samt dess uppdelning och indelning av ingående element och gränssnitten dem emellan (Kruchten, 2002). Annorlunda uttryckt ansvarar arkitekten för att de arkitekturellt intressanta komponenterna lyfts fram, organiseras och struktureras på ett sätt som utgör en hållbar konstruktion att basera systemet på. I etableringsfasen (Elaboration) har arkitekten en central roll i jämförelse med projektledaren.

4.5. Arkitekturens representation - 4+1-vymodellen[9] över arkitektur

Ett viktigt inslag i RUP är 4+1-vymodellen som har sitt ursprung i tidigare notationstekniker för objektorienterad systemutveckling som framarbetades under 80- och 90-talen vilka visade på behovet för flera olika diagram, modeller och figurer för arkitekturell systembeskrivning. Så småningom lanserades tankegångar om hur man via olika perspektiv kunde fånga in det som syntes nödvändigt. Phillipe Kruchten var dock först med att redovisa 4+1-vymodellen som den ser ut idag vilket han gjorde 95 se figur 7. (Lunell, 2003)
Eftersom arkitektur är ett komplext begrepp och svårt att klart definiera i ett svep kan det istället beskrivas ur ett flertal olika vyer vilka samordnats för att tillsammans ge en komplett bild av systemarkitekturen. Det ger att de olika intressenterna kan kommunicera sinsemellan om samma sak men utifrån sina respektive områden och perspektiv (Kruchten, 1995). Det är viktigt att representera arkitekturen på ett så bra sätt som möjligt för varje intressentgrupp så att semantiken når fram till samtliga (Kruchten, 1995). En arkitekturvy anpassar arkitekturrepresentationen efter intressentgruppen och dess villkor eftersom intressenterna har olika problem och olika fokus i systemutvecklingsprocessen. Vyerna kan tyckas separata från varandra men måste koordineras noggrant för att inte förståelseproblem ska uppstå eller helheten gå förlorad. Elementen som ingår i vyerna är nämligen inte helt åtskiljda utan kan mycket väl kopplade mellan vyerna (Kruchten,

[9] Kommer framöver ses vara synonymt med det smidigare uttrycket vymodellen

1995). Varje vy är alltså en abstraktion på en ”lagom” aggregerad nivå av ett system ur en viss intressentgrupps perspektiv.

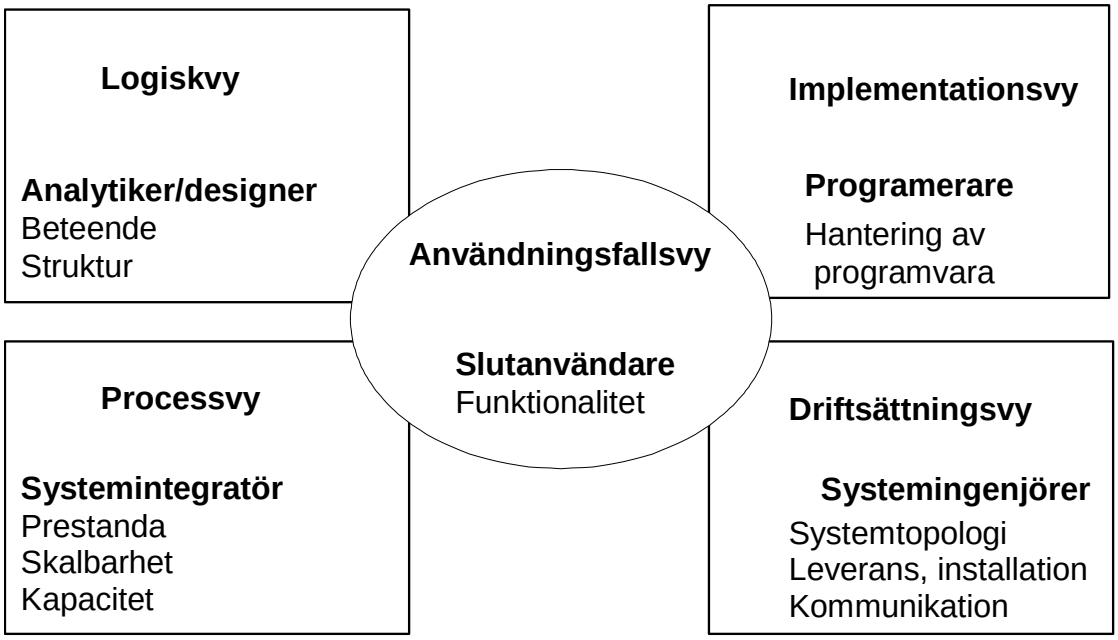
4.5.1. Vad som utgör en vy

Varje enskild vy bygger på en modell men de båda är inte identiska eftersom den underliggande modellen är mycket mer detaljerad än den vy som den ligger till grund för (Lunell, 2003). Modellens uppgift är att representera den verklighet och problemsituation som ska systemeras medan arkitekturvyns uppgift är att erbjuda ytterligare en abstraktion på arkitekturellt intressant nivå. Vyn ska lyfta fram de element som är intressanta hos den tillägnade intressentgruppen och beskriva systemets övergripande organisation samt beteende.
Eftersom de olika arkitekturvyerna representerar samma system fast ur olika intresseperspektiv måste man för varje vy klart bestämma vad som ska ingå i den vad som ska lämnas därhän eller vad som bättre hör hemma i en annan vy. Exempel på vad som kan ses vara intressant att definiera enligt Kruchten 2002:

- Vad gäller den specifika vyn måste man besluta om vilka frågor som är aktuella samt vilka intressenter som rörs av vyn.
- Vilka delar som är relevanta och därmed också ska visas i vyn inklusive relationerna dem emellan.
- Dess organisation, alltså de principer man väljer att ordna vyn efter.
- Hur vyerna och deras innehåll är relaterat till varandra.
- Vilket är det bästa sättet att producera den aktuella vyn på.

Själva modellen innehåller som namnet antyder fem olika vyer som alltså tillsammans täcker in och redogör för hela systemets arkitektur ur olika perspektiv på ett abstrakt plan. Alltför distinkta detaljer lämnas åt sidan för att behålla en god översiktsnivå vilken förtydligar viktiga karakteristiska som annars går förlorade i detaljrikedomen (RUP, 2001). Nedan visas själva figuren över 4+1-vymodellen, vilka intressenter eller områden de olika vyerna täcker in samt en kort redogörelse för dem. De vyerna som är intressanta i den aktuella situationen förs in i den för arkitekturen så viktiga artefakten arkitekturbeskrivning (Software Architecture Document). Informationen som återges i de olika vyerna är hämtade ur RUP (2001) på inrådan av Svensson (2004-05-07).

RUPs 4+1-vymodell över arkitektur



Figur 7. 4+1-Vymodellen (Kruchten, 1995).

- **Användningsfallsvy (Use Case View)**
  Det finns endast en användarfallsvy av systemet som ska åskådliggöra användningsfall och användningsscenarios vilka beskriver systemets arkitekturellt signifikanta beteende eller tekniska risker. Användningsfallsvyn kontrolleras och uppdateras i början av varje iteration i utvecklingsprocessen. Det viktiga här är att fokus stannar på systemets beteende ur ett arkitekturellt viktigt perspektiv och bygger på den underliggande Användarfallsmodellen (Use Case Model) som speglar systemets funktionalitet och beteende framtaget ur arbetsflödet för krav. Functionality i FURPS+modellen.

- **Logisk vy (Logical View)**
  Ger en bild av hur systemet är organiserat samt dess beteende ur ett implementationsoberoende designperspektiv vilken används i arbetsflödet för analys och design. Den logiska vyn illustrerar viktiga användningsfalls realiseringar, delsystem, paket, klasser vilka alla bidrar till att förverkliga systemets arkitekturellt intressanta egenskaper och beteende. Vyn grundar sig på designmodellen vilken representerar en implementationsoberoende systemlösning som tas fram i arbetsflödet för analys och design. Även den här vyn uppdateras löpande i början av iterationerna under systemutvecklingens gång.

- **Implementationsvy (Implementation View)**
  Visar hur ett system är uppbyggt av komponenter och delsystem samt hur dess källkod är arrangerad och dess huvudsakliga syfte är att fånga arkitekturellt viktiga beslut gjorda gällande implementationsspecifika frågor. Typiskt innehåll för implementationsvyn är en numrerad översikt över samtliga delsystem, komponentdiagram som visar hur alla delsystem är organiserade i exempelvis skikt eller lager och hierarkier. Termen programvaruarkitektur kommer till sin rätt i den här vyn. Användningsområden där den här vyn kan användas är vid tilldelning av implementationsspecifikt arbete till projektets individer, uppskattning av hur mycket kod som behöver generas, resonemang om storskalig återanvändning och dylikt.

- **Processvy (Process View)**
  Processvyns innehåll tjänar som en bas till förståelse av systemprocessernas organisation och tas även det fram i arbetsflödet för analys och design. Vyn illustrerar processernas sammansättning i hela systemet inklusive mappning av klasser och delsystem till specifika processer och trådar. Detta har explicit att göra med systemets exekvering och hur allt däromkring går till och planeras (eventuellt in i minsta detalj). Mycket av detta arbete görs numer mer eller mindre automatiskt av den driftsmiljö systemet ska implementeras i av operativsystem, applikationsservrar, Data Base Management Systems (DBMS) etc. Ytterligare några exempel på vad som är av intresse i vyn är responstider, genomflöden av data, feltolerans, samtidighet och skalbarhet. Också denna vy uppdateras i varje iteration för ständigt vara aktuell.

- **Driftsättningsvy (Deployment View)**
  Avser att beskriva systemets fysiska arkitektur och hur det konfigureras för att uppfylla ställda arkitekturella krav. I den här vyn berörs planeringen och organisationen av systemets drift i den fysiska miljön vilket omfattar distribution av systemprocesser och trådar på systemets fysiska noder (exekveringsmiljöer). Här åskådliggörs hur den tidigare processvyns innehåll faktiskt körs i systemets fysiska driftsmiljö.

Vyerna vänder sig till olika intressenter som är intresserade av olika saker så därför ges några exempel på vad de kan vara nedan enligt Kruchten 1995 och 2002.

- **Systemanalytikern** använder arkitekturen för att organisera och tala om kraven och förstå de teknologiska begränsningarna och riskerna.

- **Projektledaren** för programvaran använder den för att organisera projektgruppen och planera utvecklingen.

- **Designers** behöver den för att förstå de underliggande principerna och hitta gränserna för det egna designarbetet. De fokuserar också på de arkitekturellt

betydande klasserna, mekanismerna och konventionerna framför logiska detaljer hos klasserna.

- **Arkitekten** använder den för att resonera om vidareutveckling och återanvändning.

4.5.2. Arkitekturellt betydande beståndsdelar

Enligt Kruchten (2002) är en av huvudidéerna med 4+1-vymodellen ~~är~~ alltså att den ur olika perspektiv ska fokusera på arkitekturellt betydande beståndsdelar och nedan ges några exempel på vad det kan vara för något:

- Huvudklasser (specifikt de som speglar affärsobjekt).
- Arkitekturella mekanismer eller konventioner som förser klasser med beteende.
- Mönster och ramverk.
- Skikt och delsystem.
- Gränssnitt
- Processer

I nästkommande två rubriker där FURPS+modellen för infångandet, kartläggandet och klassificering beskrivs återkommer vi till hur arkitekturellt betydande komponenter och mekanismer relaterar till 4+1-vymodellen över arkitektur. Främst är det vymodellens hantering och representation av de arkitekturellt betydande komponenterna eller mekanismerna som kommer tas upp.

5. En systemarkitektur arbetas fram i RUP – kravhantering, arbetsflöden, aktiviteter och iterationsplaner

Tidigare har översikter av RUP, systemarkitekturer och de båda tillsammans presenteras men hittills har inte mycket sagts om hur en systemarkitektur arbetas fram i RUP vilket presenteras här.
I den bild (se Figur 5) som tidigare använts för att ge en översiktsblick över RUPs konstruktion i två dimensioner visades bland annat nio primära arbetsflöden eller discipliner som de också kallas. Det är här på sin plats att närmare beskriva arbetsflödenas roll i utvecklingsprocessen eftersom deras utförande, via detaljer, aktiviteter, iterationsplaner och roller, direkt medverkar till någon typ av mätbart och betydande resultat, exempelvis systemarkitekturens framkomst. Alla arbetsflöden behandlas inte eftersom fokus främst läggs på de delar av utvecklingsprocessen som på ett mer direkt sätt har att göra med systemarkitektur. Antingen att de, deras komposition och utförande direkt medför till arkitekturens konstruktion eller att på annat sätt antingen direkt bidrar till den eller är beroende av den. Av detta följer att de mest intressanta arbetsflödena är krav (Requirements) i förberedelsefasen (Inception) tillsammans med analys och design i etableringsfasen (Elaboration).

Först behandlas FURPS+modellen i arbetsflödet för krav i och med att en arkitekturell kravbild måste utgöra grunden för arbetet med arkitektur. Därefter förklaras arbetsflödenas roll i framställningen av en systemarkitektur tillsammans med aktiviteterna arkitekturell analys (Architectural Analysis) och förfining av arkitekturen (Refine Architecture). Slutligen visas ett exempel på hur en projekttillämpning kan se ut för framtagning av en arkitekturprototyp i en typisk iterationsplan.

5.1. Framtagandet av en arkitekturell komponent

Att hantera krav i RUP handlar om att identifiera dem, dokumentera, organisera och lagra dem samt att kontinuerligt följa deras förmodade förändringar (Lunell, 2002).
Det är viktigt att inse att en systemarkitektur inte uppkommer av sig självt eller är någon typ av efterhandskonstruktion utan att dess framställning precis som systemfunktionalitet har sitt ursprung i en grundläggande kravbild (Svensson, samtal 2004-03-17). Det är nödvändigt att man verkligen fångar de arkitekturella kraven och löser dem med hjälp av en utformning av systemarkitekturen på ett sätt som minimerar risken att någonting betydande förbises eller negligeras för att resultatet, arkitekturen, ska bli bra. I detta syfte förespråkas användningen av IEEEs[10] FURPS+modell för att fånga och klassificera arkitekturella krav på ett systematiskt sätt (Eeles, 2001). Arbetet med modellen har sin plats inom arbetsflödet för krav (Requirements, se Figur 5) (RUP, 2001) som också är det arbetsflöde där de övriga funktionella systemkraven fångas och kartläggs. Det är alltså tidigare än aktiviteten Architectural Analysis som tar sin början i arbetsflödet för analys och design i etableringsfasen och bygger på vad FURPS+ genererat. Då det är det

[10] Institute of Electrical and Electronics Engineers

lämpligt att börja arkitekturutvecklingen med fångst och kartläggning av arkiteturella krav är det också lämpligt att visa hur RUP definierar begreppet krav. I RUP, V. 2001A.04.00 finns följande att finna i dess ordlista: ”A requirement describes a condition or capability to which a system must conform; either derived directly from user needs, or stated in a contract, standard, specification, or other formally imposed document”. Definitionen avser krav i största allmänhet så det finns naturligtvis utrymme för mer specifika beskrivningar av olika kravtyper.

5.1.1. Kravfångst med hjälp av FURPS+

Vad som är av relevans här är arkitekturellt intressanta krav vars definition kan delas upp i implicita respektive explicita arkitekturkrav (Eeles, 2001). Implicita arkitekturkrav hittas vid kartläggning av attribut associerade till ett systemkrav och är därför ofta relaterade till direkta affärsbehov, exempelvis kan högriskkrav, högt prioriterade eller destabiliserande krav vara av arkitekturell betydelse (Eeles, 2001). Explicita arkitekturkrav är de som tydligt identifieras som arkitekturellt intressanta och kan ofta vara tekniska till sin natur (Eeles, 2001). Exempel på explicita arkitekturkrav kan gälla specifika driftsmiljöer, åtkomst som MTBF (Mean Time Between Failure), 24/7 etc., språkanpassningsmöjligheter med mera. Den här typen av krav kan variera stort och en del är funktionella medan många är icke-funktionella till sin natur, en del rent tekniska och mekanismspecifika, andra inte. Ett systematiskt sätt att kartlägga arkitekturkraven är alltså FURPS+. Bokstäverna i modellnamnet är förkortningar och står för Functionality, Usability, Reliability, Performance, Supportability och plustecknet står för begränsningar i form av design-, implementations-, gränssnitts- och fysiska krav. Följande punkter och exempel klassificerar alltså olika arkitekturella krav efter respektive kravs kännetecken.

- **Functionality**
  Representerar produktens generella funktionalitet och kan exempelvis beröra systemets utmärkande drag i form av beteende och säkerhet (Eeles, 2001). De funktionella kraven beskriver det som användarna ska kunna utföra med hjälp av systemet och i de flesta fall är systemen slutna i den mening att användarna inte kan lägga till nya funktioner vartefter behov dyker upp. Uttrycks inte ett nödvändigt behov i kravform kommer följaktligen inte den funktionen realiseras och finnas tillgänglig. Vidareutveckling består till stor del av att anpassa eller tillföra funktionalitet vilket ställer krav på arkitekturens konstruktion. (Lunell, 2003)

- **Usability**
  Omfattar karaktäristika som estetik och kontinuitet i användargränssnittet och omfattar därmed kvaliteter som kan ses kopplade till mänskliga faktorer (se punkten Anvädningsfallsdriven process under rubriken Övriga nyckelegenskaper) samt dokumentations material, manualer, wizards, tutorials med mera. (Eeles, 2001) Annorlunda uttryckt innebär användbarhetskraven att systemet ska vara lättanvänt vilket betyder att det finns bra utbildningsmaterial, att systemet är hjälpsamt och tilltalande till sin utformning. Användbarhetskrav är svåra att

avgöra när de är tillgodosedda och här finns speciella metoder att tillgå. (Lunell, 2003)

- **Reliability**
  Berör främst begrepp som åtkomst, tillförlitlighet, integritet och feltolerans. Här kommer RUPs riskkonfronterande egenskaper in och gör sig användbara på ett konkret sätt. Vad som är aktuellt här är saker som förutsägbarhet, Mean Time Between Failure (MTBF), återhämtningsbarhet, träffsäkerhet, validitet och reliabilitet. (Eeles, 2001) Ovan punkter syftar till att göra systemet så tillförlitligt som möjligt och mycket hänger på hur det hanterar undantag som fel samt hur väl det klarar av en krasch och hur det minimerar de negativa konsekvenserna en sådan händelse medför. Även hur förutsägbara konsekvenserna av ett fel är spelar in. (Lunell, 2003)

- **Performance**
  Har att göra med genomströmning, responstid, återhämtningstid från fel, tid till driftsklar samt nedstängningstid och påverkar därmed direkt systemfunktionalitet där prestandaparametrar är av vikt (Eeles, 2001). Vad som är prestanda i sammanhanget avgörs till stor del av den aktuella tillämpningen, det vill säga vad dess huvudsakliga syfte är och hur prestandakrav ställs utifrån det. Det är exempelvis skillnad på prestandakraven hos ett transaktionssystem och en beräkningsintensiv simuleringsapplikation. Ofta definieras prestandakrav i mätvärden. (Lunell, 2003)

- **Supportability**
  Berör testbarhet, underhållsnivå, kompabilitet, konfigurerbarhet, förändringsbarhet, installerbarhet, skalbarhet och mobilitet (Eeles, 2001). Vilka krav som ställs på underhållbarhet bestäms mycket av hur produkten och dess omgivning förväntas förändras med tiden. Alla program ska konstrueras så att de tillåter förändring, anpassning och utvidgning efter hur gamla behov förändras eller nya uppstår. Det kan handla om språk såväl som kompabilitet med äldre program och system.(Lunell, 2003)

+ **Begränsningar**
  Handlar ofta om designbegränsningar vilka direkt berör integration av nyutvecklade (del)system i befintliga miljöer där infrastrukturen som redan används kan sätta begränsningar för informationslagring i en viss typ av databas (Wessberg, kommentar 2004-05-23).

  - **Designkrav**
    Fungerar ofta som designbegränsningar när det specificerar eller avgränsar olika designmöjligheter för systemet. Det kan exempelvis röra sig om systemets grundstruktur, om det ska vara distribuerat, skiktat, client-server eller hur dess lagringsegenskaper ska se ut till exempel om den ska utgöras av

en relationsdatabas eller dylikt. Exemplena är alla olika typer av begränsningar som designen måste ta hänsyn till.

- **Implementationskrav**
  Berör främst begräsningar gällande systemets kodning eller konstruktion och täcker frågor om val av språk och standarder för kodning programvara. Andra exempel kan röra sig om vilken utvecklingsmiljö som ska användas i konstruktionen av systemet och vilka krav det för med sig.

- **Gränssnittskrav**
  Specificerar en extern entitet som systemet måste interagera med och då berörs frågor som kommunikationsprotokoll, säkerhet- och rättighetsaspekter, tekniska kommunikationskrav med mera. Allt som kan tänkas påverka eller påverkas av kommunikationskraven mellan systemet och någonting annat.

- **Fysiska krav**
  Val och krav som påverkar den fysiska miljön som krävs för att systemet ska kunna driftas korrekt. I den fysiska miljön som påverkas ingår främst den hårdvara som ska tjäna som plattform för systemet.

Förespråkare av FURPS+modellen menar på att det behövs ett distinkt och systematiskt sätt att fånga, kartlägga och klassificera den här typen av krav eftersom de kan vara svårfångade och samtidigt i vissa fall till och med viktigare än systemets funktionella krav (Eeles, 2001). De krav FURPS+ behandlar är viktiga ur ett mycket brett och heltäckande systemperspektiv, på en hög nivå och är därför av stor vikt när systemets grundstomme, det vill säga arkitekturen, ska konstrueras. Eeles påstår också att en av anledningarna till att de arkitekturellt signifikanta kraven ofta förbises och att framtagandet av systemarkitekturen i övrigt inte får den plats den förtjänar bland annat beror på svårigheter att hitta de krav som lägger grunden för utformandet av arkitektur. Mer om detta senare.
När väl FURPS+modellen tillämpats i utvecklingsarbetet med att fånga och kartlägga systemarkitekturella frågor och krav har man erhållit en utgångspunkt, en bas, för att designa och realisera lösningar på de arkitekturella kraven och i sin förlängning organisera dem till en konkret systemarkitektur. De generella resultat som man erhåller efter tillämpningen av FURPS+ i arbetsflödet för krav dokumenteras i artefakten kompletterande specifikationer (Supplementary Specification) medan de som berör specifika användningsfall dokumenteras i respektive användningsfallsdokumentation (Wessberg, kommentar 2004-05-23). Dokumentationen av FURPS+kraven ska bland annat presentera kvalitetsattribut för systemet inklusive användbarhet (Usability), pålitlighet (Reliability), prestanda (Performance) samt andra krav och begräsningar som kan utgöras av plustecknet i FURPS+ (RUP, 2001).
För att i det här läget gå vidare med framtagandet av en arkitektur använder man ofta arkitekturella mekanismer för att erbjuda lösningar på arkitekturkrav (Eeles, 2001). Under följande rubrik kommer mekanismer att beskrivas generellt samt mer specifikt som olika typer av mekanismer.

5.1.2. Mekanismer

En mekanism är generellt sett bestående av samverkande komponenter vars syften varierar efter den aktuella situationen de konstruerats för. Främst är deras uppgift att förenkla något genom att mer eller mindre automatisera eller schablonera uppgiftens utförande (Kruchten, 2002). Det finns flera typer av mekanismer till exempel analys och designmekanismer eller så kan det handla om tillägg av nya systemkomponenter i samband med utbyggnad där mekanismer används för att på ett enklare och mer konsekvent sätt klara uppgiften än om det skett på ett mer godtyckligt sätt (Lunell, 2003).

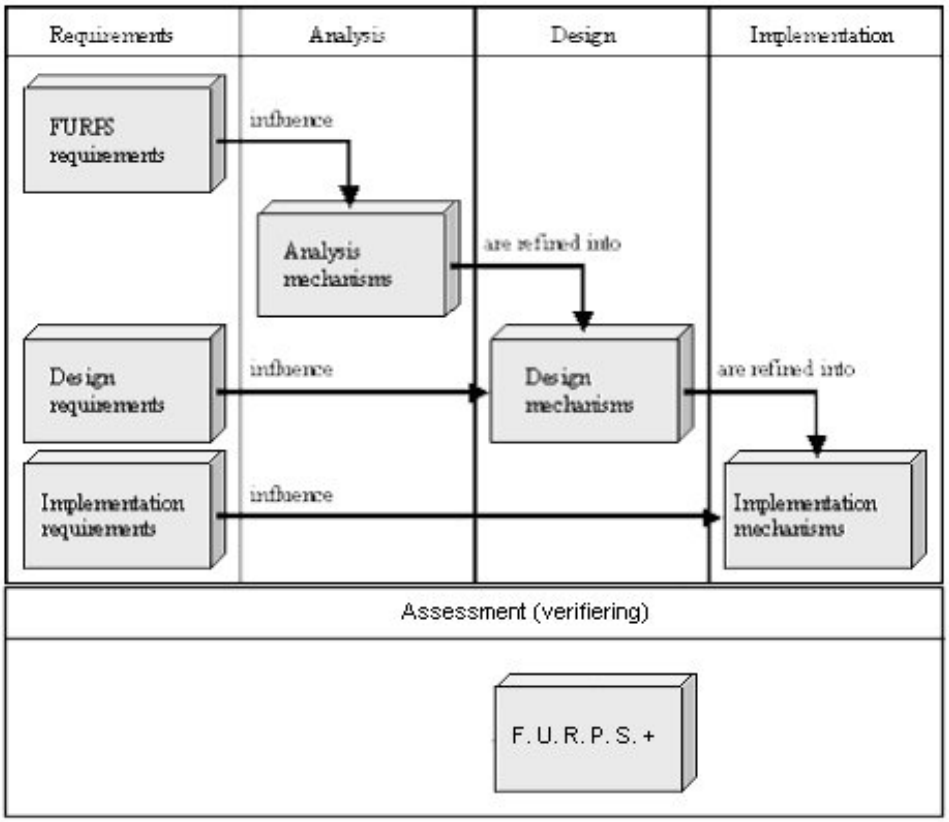
- **Arkitekturell mekanism**
  En klass, grupp av klasser eller ett mönster som utgör en lösning på ett frekvent förekommande problem, ett snabbt sätt att tillämpa en eftersökt problemlösning. Deras närvaro bidrar till att ge en klass eller dylikt ett beteende eller egenskap. Främst återfinns de i mellanliggande och lägre arkitekturella skikt. (RUP, 2001)

- **Analysmekanism**
  En generell mekanism som representerar en implementationsoberoende lösning av ett problem. En abstraktion som påvisar en konceptuell lösning. (Eeles, 2001)

- **Designmekanism**
  En vidareutvecklad analysmekanism som generellt tar hänsyn till implementationsmiljön den ska finnas i men uttrycker fortfarande inte en explicit beskrivning av den. (Eeles, 2001)

- **Implementationsmekanism**
  Är en förfinad designmekanism som exakt specificerar implementationen av mekanismen (Eeles, 2001).

I figur 8 nedan kan ett samband mellan kravfångst enligt FURPS+modellen, övriga design och implementationskrav samt arbetet med analys-, design- och implementationsmekanismerna åskådliggöras på ett överblickbart sätt.



Figur 8. Illustrerar relationen mellan krav fångade och kartlagda i FURPS+, lösande mekanismer samt verifiering av desamma (Eeles, 2001).

Det kan vara lämpligt att återgå till figur 7, för att se hur den tidigare redovisade 4+1-vymodellen anknyter till att hantera och representera de arkitekturellt viktiga mekanismerna och komponenterna av olika slag. Man kan i huvudsak se vilka olika typer av mekanismer och beståndsdelar de olika vyerna omfattar samt för vilka intressenter respektive vy och innehåll är ämnade. Ett exempel som kan ges är att den logiska vyn riktar sig till analytiker och designers samt omfattar arkitekturellt signifikanta mekanismer och komponenter som utgör systemets struktur. Varje vy innehåller olika UML-diagram avsedda och anpassade för de olika intressentgruppernas arbets- och ansvarsområden samt kognitiva förmåga. Resultatet som FURPS+ genererar används som input i senare del av arbetet med framtagning av arkitekturen.

5.1.3. Verifiera arkitekturen med hjälp av FURPS+

När grunden för arkitekturellt signifikanta krav och mekanismer är lagd måste den utvärderas och i RUP utgör FURPS+ även möjlighet till en kvalitetssäkring när det exempelvis gäller att verifiera den arkitektur man konstruerar samt dess grundläggande krav. Man återgår till de krav som initialt ställdes på arkitekturen och som man fångade, klassificerade och löste genom arbete med FURPS+ (Svensson, intervju 2004-05-07). De typer av tester som utförs på systemarkitekturen gentemot FURPS+ resultaten är följande enligt RUP, 2001:

- **Functional testing**
  Den här typen av test verifierar att systemet exekverar specificerade användningsfall på korrekt sätt. Funktionella tester kan inkludera featurebaserad testning, användningsscenarier och säkerhetsfunktioner.

- **Usability testing**
  Utvärderar produkten ur användarens synvinkel och testar den ur mänskliga perspektiv där gränssnitt, korrekthet, kontinuitet, tillgänglighet, dokumentation, hjälpmedel och utbildningsmaterial är av intresse.

- **Reliability testing**
  Säkerställer att produkten tillgodoser de pålitlighetskrav som ställts på den, det vill säga att bland annat produktens feltolerans utvärderas med hjälp av exempelvis stresstester där produkten utsätts för påfrestningar för att kontrollera huruvida den håller kvalitetsmåttet gällande pålitlighet eller inte. Övriga tester här är integritets, struktur och volymtester.

- **Performance testing**
  Avser att verifiera om produkten klarar att tillmötesgå de krav som ställs på dess prestanda i avsedd driftsmiljö. Tester här omfattar benchmark- och intensivitetstester.

- **Supportability testing**
  Avgör om produkten kommer klara av att driftsättas enligt givna krav och förväntningar, här inkluderar man olika installations och konfigurationstester.

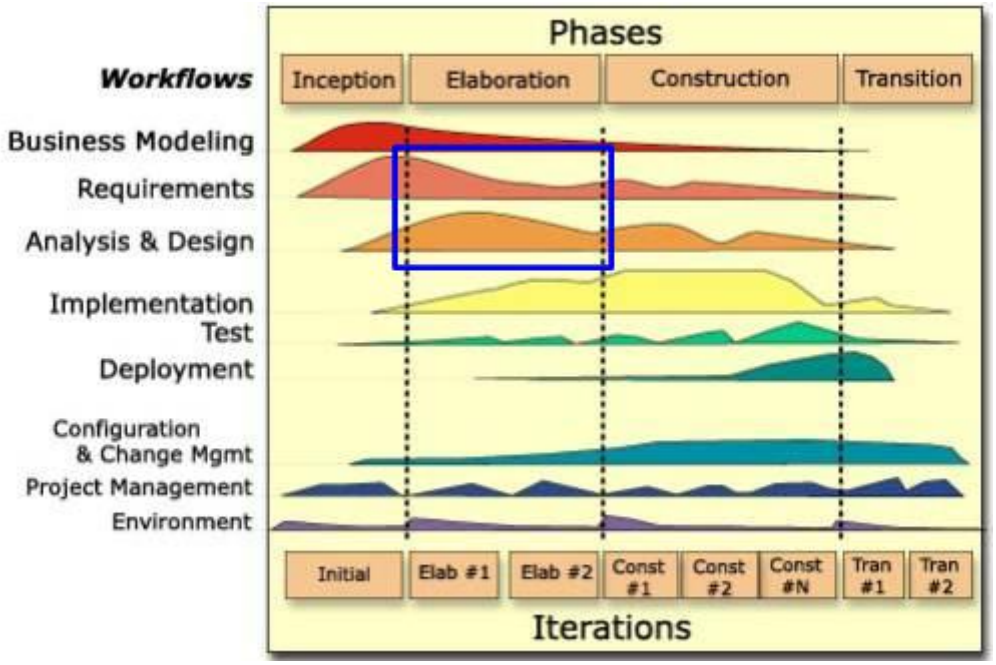
I figur 8 ovan illustreras hur arbetet under kravhanteringsprocessen verifieras (Assessment) fortlöpande.
Från arbetsflödet för krav tar man sig i det arkitekturella utvecklingsarbetet vidare till arbetsflödet för analys och design i etableringsfasen. Under nedanstående rubrik beskrivs hur RUPs statiska struktur används för att bilda den dynamiska utvecklingsprocessen som man faktiskt utvecklar efter.

5.2. Statiska arbetsflöden, detaljer och aktiviteter

Enklast uttryckt kan arbetsflöden beskrivas som sammansatta sekvenser av aktiviteter som producerar ett resultat – ett mätbart värde av en sammansatt insats (Kruchten, 2002). Anledningen till att man lägger ihop flera aktiviteter i logiska grupper är att värdet av en enskild aktivitet utförd av en rollinnehavare inte är mycket mer än marginellt i det stora hela (Lunell, 2003). Arbetsflöden kan på ett överblicksgivande sätt beskrivas på flertalet sätt men med hjälp av aktivitetsdiagram enligt UML-notation följer man RUPs kompabilitet med just det objektorienterade notationsspråket. Mer formellt utgörs arbetsflöden av arbetsflödesdetaljer som består av logiskt grupperade aktiviteter,

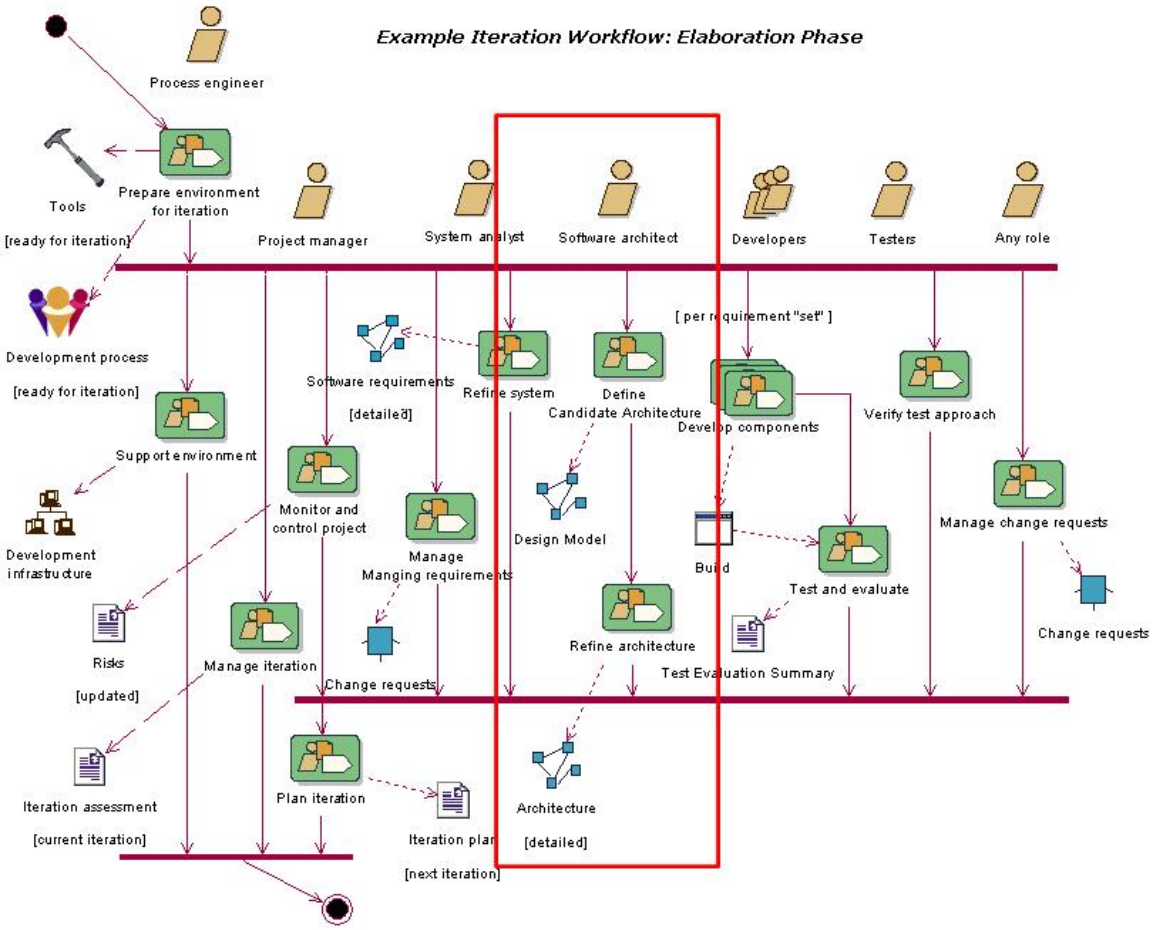
artefakter och dylikt. Det är här en omvandling från statiska element till den dynamiska systemutvecklingsprocessen blir påtaglig (Svensson, samtal 2004-05-19).

- **Primära arbetsflöden (Workflows)**
  Exempelvis är arbetsflödet för krav (Requirements) en högnivåbeskrivning av vad som kan eller bör ske inom just det arbetsflödet. Arbetsflödena generellt delar in de olika aktiviteterna, rollerna och artefakterna i logiska grupper för olika arbetsområden. De primära arbetsflödena är de nio som avbildas i modellen över RUPs tvådimensionella uppbyggnad. Som översiktsbild kan de tyckas få processen att se ut som en vattenfallsprocess men utformningen av arbetsflödesdetaljer och iterationsplaner talar mot detta. De primära arbetsflödena är grundarbetsflöden vars huvudsakliga syfte är att erbjuda en översikt av *alla* aktiviteter (Kruchten, 2002), inte nödvändigtvis de som man väljer för aktuellt projekt. För att återknyta till tidigare del av uppsatsen handlar de primära arbetsflödena eller disciplinerna om RUPs statiska struktur eller annorlunda uttryckt de element processen erbjuder för framtagandet av situationsberoende projekttillämpningar. I figur 9 nedan markeras de två i uppsatsen huvudsakligen berörda arbetsflödena i den fas de primärt undersökts.

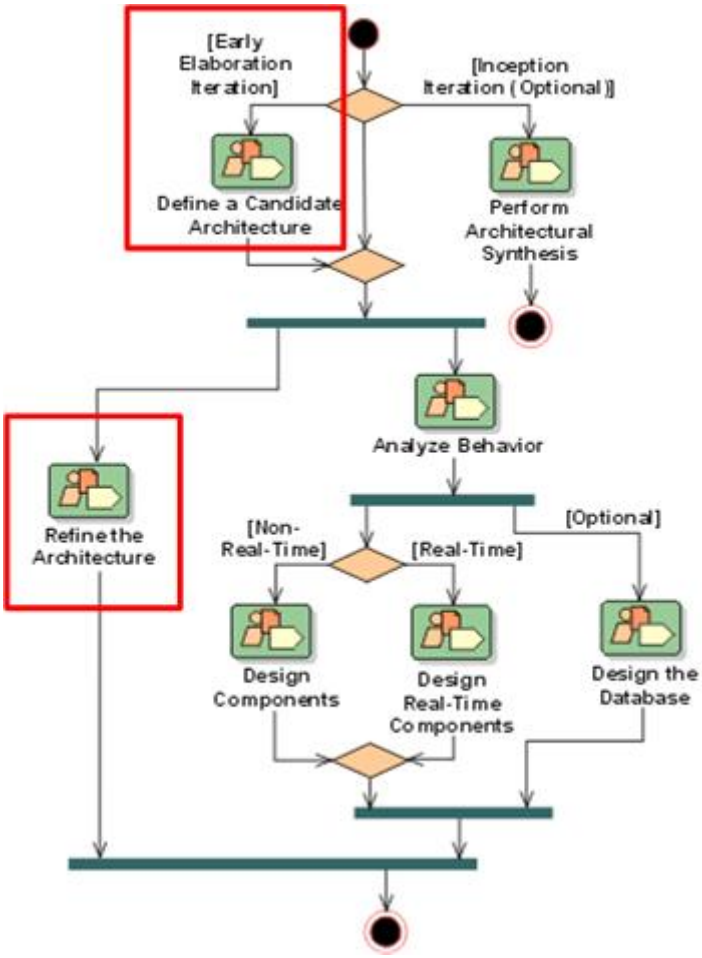


Figur 9. Modell på RUPs övergripande tvådimensionella struktur, lägg i detta läge främst märke till de primära arbetsflödena (exempelvis krav (Requirements) och analys och design) (RUP, V. 2001). Markerade element visar för sammanhanget (och uppsatsen) intressanta delar.

- **Arbetsflödesdetaljer (Workflow details)**
  Berör hur man på ett typiskt sätt kan hantera ett problem av någon typ som exempelvis definitionen av en kandidatarkitektur (Define Candidate Architecture) eller förfina arkitekturen (Refine Architecure) som ses markerade i figur 10 och 11 nedan. Detaljer som i sin tur består av aktiviteter används för att bryta ner de mycket omfattande primära arbetsflödena till mindre specifika grupper av nära relaterade aktiviteter. Här beskrivs också de artefakter som är knutna till de olika aktiviteterna och tjänar som in- respektive utdata. Ett i sammanhanget bra exempel på arbetsflödesdetalj är definitionen av en kandidatarkitektur.

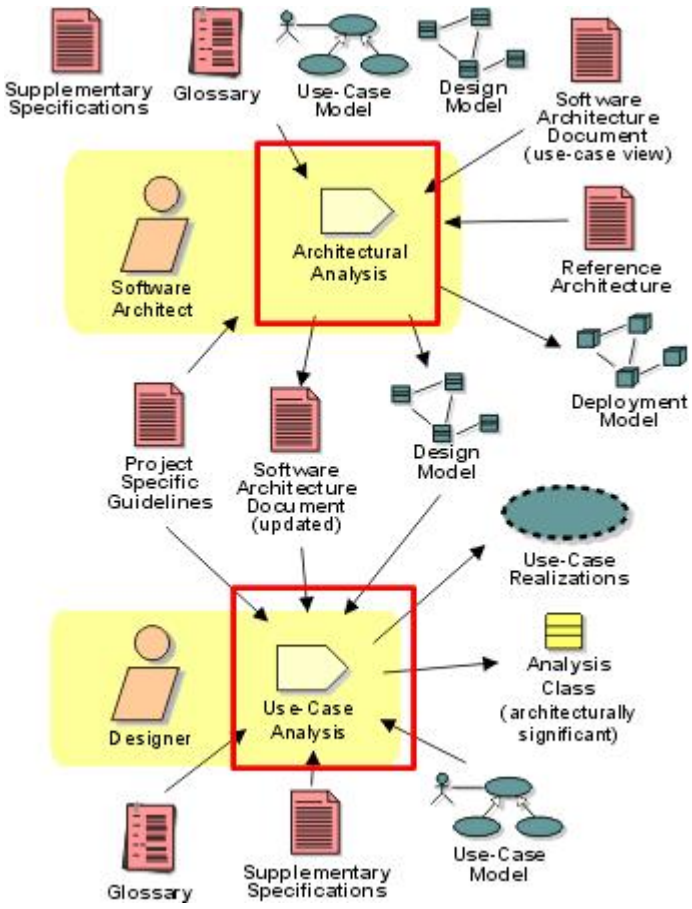


Figur 10. Visar en typisk arbetsflödesdetalj inom etableringsfasen (Elaboration) och ger ett perspektiv på var i den delen av processen arkitekturen har sin plats (RUP Evaluation Assembly V2003).



Figur 11. Exempel på arbetsflödet för Analys & design ur Elaborationfasen i ett aktivitetsdiagram (RUP, 2001). De rödmarkerade elementen i figuren är arbetsflödesdetaljer.

I figur 12 nedan visas arbetsflödesdetaljen Define a Candidate Architectures innehåll där aktiviteterna arkitekturell analys (Architectural Analysis) och användningsfallsanalys (Use-Case Analysis) dominerar.



Figur 12. Bild över arbetsflödesdetaljen Define a Candidate Architecture som visar de två centrala aktiviteterna (i röda markeringar), hur de relaterar till varandra samt vilka in-respektive outputs de båda har (RUP Evaluation Assembly V2003).

Sammanfattningsvis utgörs alltså RUPs vertikala dimensionen av arbetsflöden sammansatta av arbetsflödesdetaljer vilka i sin tur är hopsatta av enskilda aktiviteter utförda av roller. Det är viktigt att framhålla att arkitekturen i sig inte är ett enskilt arbetsflöde utan ett resultat av utförda aktiviteter inom arbetsflödet för krav samt analys och design. Det finns alltså tre nivåer i den statiska elementstrukturen: arbetsflöden, arbetsflödesdetaljer och aktiviteter medan den faktiska tillämpningen uttrycks i iterationsplaner (Svensson, samtal 2004-05-19).

5.2.1. Arbetsflöden för framtagning av arkitektur

Arbetsflöden för framtagning av en arkitektur är situationsanpassade projekttillämpningar av RUPs statiska element. Man tittar på en anpassad lösning för en uppgift eller detalj (här konstruktion av arkitektur) i en viss situation genom valet av aktiviteter. (Svensson, samtal 2004-05-19)
När man erhållit de arkitekturellt signifikanta kraven och därefter letat efter möjliga lösningar på dem med hjälp av olika typer av mekanismer, främst arkitekturella

mekanismer, samt utvärderat resultatet gentemot ursprungliga FURPS+ är det dags att börja arbetet med arkitekturkonstruktionen på analys- och designnivå och då övergår arbetet med arkitekturen i en ny fas och ett annat arbetsflöde (Svensson, samtal 2004-05-18). Från förberedelsefasen (Inception) förflyttas arkitekturarbetet till den nästkommande etableringsfasen (Elaboration) samt att arbetsflödet övergår till analys och design och aktiviteterna arkitekturell analys (Architectural Analysis Acitivty) samt arbetsflödet förfina arkitekturen (Refine Architecture).
Följande material är hämtat ur Phillippe Kruchtens bok The Rational Unified Process och representerar de två viktigaste arbetsflödesdetaljerna vars tillämpning direkt syftar till framtagandet av en systemarkitektur. Varje arbetsflödesdetalj består av flera olika aktiviteter. Denna rubrik relaterar till punkten Arbetsflödesdetaljer på sidan Y.

Arbetsflödesdetalj: Definiera kandidatarkitektur (anlysbit)

Arbetsflödesdetaljen består av aktiviteterna

- Arkitekturell analys, utförs av arkitekten
- Användningsfallsanalys, utförs av designern

Syftet med arbetsflödesdetaljen är att:

- Skapa ett första utkast till systemarkitekturen genom att definiera:
  - En första uppsättning arkitekturellt signifikanta element som grund till analysen
  - En första uppsättning analysmekanismer
  - En första skiktning och organisation av systemet och användarfallsrealiseringar (implementationsmekanismer) som ska hanteras i nästkommande iteration
- Identifiera analysklasser från de arkitekturellt signifikanta användningsfallen (aktiviteten Use-Case Analysis, se Figur 13)
- Uppdatera användningsfallsrealiseringarna analysklassernas interaktion

Arbetsflödesdetalj: Förfina arkitekturen (designbit)

Arbetsflödesdetaljen består av aktiviteterna

- Identifiera designmekanismer
- Identifiera designelement
- Beskriv runtime-arkitekturen
- Beskriv distribution

alla aktiviteterna ovan utförs av arkitekten.

Syftet med arbetsflödesdetaljen är att:

- Utgöra en naturlig övergång från analys- till designaktiviteter genom att identifiera:
  - Lämpliga designelement från analyselementen och
  - Lämpliga designmekanismer från relaterade analysmekanismer.
- Se till at arkitekturen fortsätter vara korrekt genom säkerställandet av att:
  - Nya designelement som identifierats i nuvarande iteration integreras med redan existerande designelement och
  - Att man uppnår maximal återanvändning av tillgängliga komponenter och designelement så tidigt som möjligt i designarbetet.

- Beskriva hur systemets runtime- och driftsättningsarkitektur är organiserad.
- Organisera implementatonsmodellen för att få en skarvlös övergång mellan design och implementation.

5.3. Dynamisk projekttillämpning av statiska strukturelement

Iterationsplaner är än mer tillämpningsorienterade än arbetsflödesdetaljer och är mer den konkreta planering för vad som ska utföras, när, var, hur, av vem, varför samt hur många gånger. Den är menad att vara så pass detaljerad att den ser ut som en konventionell projektplan och det är också bland annat här RUP öppnar upp för mer specifika projektstyrningsmetoder (Lunell, 2002). En iterationsplan beskriver alltså vad som ska hända i den aktuella iterationen på en detaljerad nivå samt fördelar aktiviteter på roller och roller på personer.

5.3.1. En typisk iterationsplan för skapandet av en arkitekturell prototyp

Nedan följer vad som kan kallas ett typiskt exempel på hur en tidig iteration i etableringsfasen kan se ut. Exemplet går ut på att redogöra för hur man bygger en arkitekturell prototyp och förutsätter att tidigare förberedelsefas (Inception) är avslutad och att dess livscykelmål är uppnått. Exemplet är hämtat från Kruchtens bok The Rational Unified Process, En introduktion samt kontrollerad i RUP-dokumentationen. Det är viktigt att påpeka att de punkter som beskrivs i iterationsplanen knyter an till de arbetsflödesdetaljer och aktiviteter som tidigare beskrivits under rubriken Arbetsflöden för framtagning av arkitektur. Beroende på hur tidig eller sen iterationsplanen är appliceras de två beskrivna arbetsflödesdetaljerna och dess innehåll i olika utsträckning på punkterna nedan, i första skedet är det arbetsflödesdetaljen definiera en kandidatarkitektur som dominerar för att i nästkommande iterationer ersättas av detaljen förfina arkitekturen och dess ingående aktiviteter (Svensson, samtal 2004-05-19).
De delar som markeras ”---” är element som inte har setts direkt nödvändiga att redogöra men deras avsaknad markeras likväl.

- Första iterationen inom **Etableringsfasen**.
    - Ta fram en plan hur grundarkitekturen ska tas fram, se arbetsflödesdetaljen definiera kandidatarkitektur (Define a Candidate Architecture).
- Andra (tidig) iteration inom **Etableringsfasen**.
    - **Projektledning:**
      Beskriv iterationsplanen, riskerna och de arkitekturella målen i stort.
        - Förslag till utvärderingskriterier för arkitekturen tas fram i diskussion och samtycke med arkitekten eller arkitektgruppen.
        - Arkitekturella risker som ska minimeras beaktas och hamnar i artefakten Risklista. Ett av målen i etableringsfasen är att lägga grunden till en stadig arkitektur.
        - Allt i stora drag.

- **Krav:**
  Avgör vilka användningsfall och andra scenarier som ska användas för att driva arkitekturutvecklingen. Se beskrivning av FURPS+ för en återkoppling till arkitekturell kravfångst och kravbearbetning.
    - Projektledaren och arkitekten diskuterar en initial användningsfallsvy för att besluta vilka användningsfall som ska hanteras i det (fortfarande) grundläggande arbetet med att ta fram en systemarkitektur. Här blir arbetsflödesdetaljen definiera kandidatarkitektur aktuell igen.
    - Arbetet här kan påverka interationsplanen så den behöver förnyas.
- ---
- Ompröva användningsfall och risker
    - Arkitekten kontrollerar genom användningsfallsvyn och tittar på nya användarfallsbeskrivningar och eventuellt en ny struktur för hur användarfallsmodellen bäst representerar dem. Det urval av användarfall som kommer vidareutvecklas genom fortsatt analys, design och implementation bestämmer systemets arkitektur.
- **Analys och design**:
  Hitta uppenbara klasser, gör en första delsystemindelning samt granska användningsfallen i detalj.
    - För att föreslå en delsystemindelning tittar arkitekten bland annat på vilka klasser som kan behövas, gällande systemkrav, ordlistan (en artefakt), användningsfallsvyn, utvecklingsgruppens generella domänkunskap. Det åligger även arkitekten att identifiera analysmekanismer som ska utgöra lösningar till vanligt förekommande analysproblem. Arkitekturbeskrivningen påbörjas (artefakt: Arkitekturbeskrivning).
      En grupp designers arbetar parallellt med arbetet att hitta klasser och/eller objekt för användarfallen eller scenarierna, eventuellt tillsammans med arkitekten.
- **Analys och design:**
  Förfina samt homogenisera klasser och hitta de som är arkitekturellt betydande. Granska resultatet.
    - Designers förfinar arbetet från förra steget och detaljplan för hur klasserna ska utnyttja analysmekanismerna tas fram. Arkitekten bestämmer vilka klasser som är intressanta ur arkitekturellt perspektiv och etablerar dem i den logiska vyn (Artefakt: Arkitekturbeskrivning).
- **Analys och design:**
  Gör designpaketindelning.
    - För att betona beteendeaspekten hos arkitekturen slår arkitekten ihop några klasser till designpaket och relaterar dem till efter relevans tills respektive delsystem.

- **Analys och design:**
  Anpassa implementationsmiljön, slå fast designen av nyckelscenarier och etablera formella klassgränssnitt. Granska resultatet.
    - Arkitekten förfinar arkitekturen ytterligare genom att härleda designmekanismer ur tidigare upprättade analysmekanismer. Kraven på designmekanismerna handhas av arkitekten som uppdaterar den logiska vyn på lämpligt sätt för att omfatta dessa. De framtagna designartefakterna granskas.
- **Analys och design:**
  Uppmärksamma tidsmässigt samtidighet och distribution i arkitekturen.
    - Samtidighet och distribution undersöks genom att arkitekten tittar på de rutiner och processer som behöver utföras samt den fysiska miljön i form av nätverk, processorer och annat. Som underlag används användningsfallen i egenskap av samverkande objekt i interaktionsdiagram, dvs. användarfallsbeskrivningsrealiseringarna (Artefakt: Arkitekturbeskrivning).
- **Analys och design:**
  Undersök den arkitekturella designen.
    - Arkitekturen evalueras.
- **Implementation:**
  Den fysiska paketeringen av arkitekturen ses över.
    - Arkitekten undersöker arkitekturdesignens implikationer på implementationsmodellen och definierar upp implementationsvyn.
- ---
- ---
- ---
- ---
- **Test:**
  Evaluera den exekverbara arkitekturen.
    - När systemets delar och komponenter har kopplats samman och integrerats, vilka de är och vad de gör definieras av den aktuella iterationens mål, testas systemet av systemtestaren. Resultaten analyseras för att utreda huruvida aktuella mål eller milstolpar har uppfyllts. Arkitekten kontrollerar detta mot de riskbedömningar som gjordes i de mera initiala skedena.
- ---

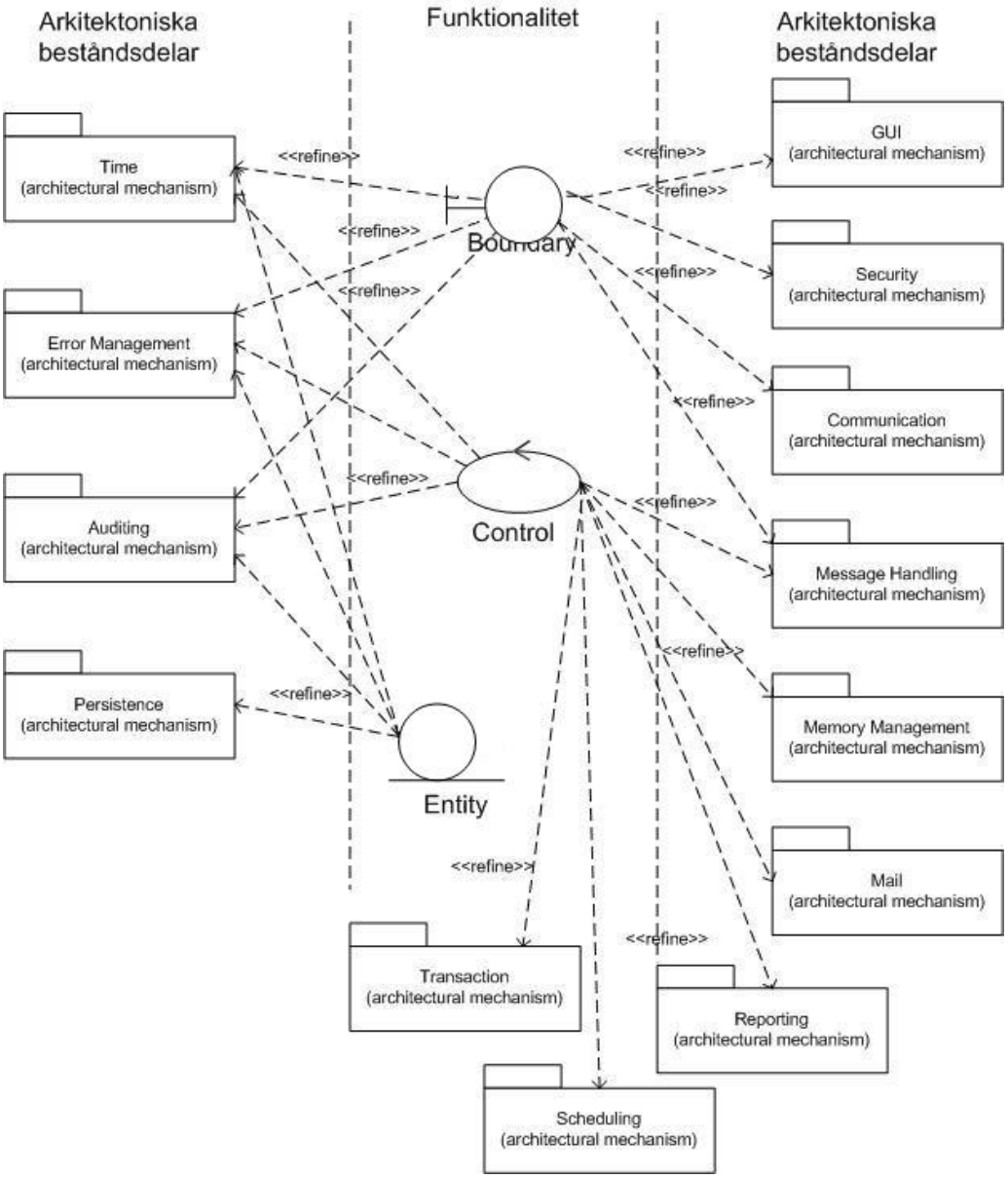
(Kruchten, 2002 och RUP, 2001)

6. Sammanfattning

Uppsatsen har vid det här laget beskrivit systemutvecklingsprocessen RUP och begreppet systemarkitektur var för sig men även tillsammans. Arbetet har redogjort för de två ämnesområdens inverkan på varandra, deras ömsesidiga beroendeförhållande samt påvisat hur en systemarkitektur arbetats fram inom RUPs statiska såväl som dynamiska ramverk och därmed redogjort för dess implikationer för en systemutvecklingsprocess och dess resultat.

Ambitionen har varit att följa en ”röd-tråd” genom att på ett logiskt sammansatt sätt visa hur arbetet med arkitekturen tar sin början i arbetsflödet för krav i förberedelsefasen (Inception) med kravfångst, kravanalys, lösande mekanismer samt utvärdering för att sedermera fortsätta i arbetsflödet för analys och design med ett mer påtagligt analys respektive designarbete med arkitekturkonstruktionen.

Den senare delen av uppsatsen har jämfört med inledande delar varit mer detaljerad och tungrodd till sin natur eftersom abstraktionsgraden sänkts för att påvisa nödvändiga delar vilka annars inte synliggjorts. För att på ett sammanfattande vis illustrera mycket av vad den tyngre delen av uppsatsen vill framhäva har figuren nedan valts då den åskådliggör förhållandet mellan ”klassisk” systemfunktionalitet i form av analysklasser och mer moderna arkitekturella mekanismer. I figuren kan man se hur de arkitekturella mekanismerna bidrar med ett antal icke-funktionella krav till den övriga systemfunktionaliteten som visas som analysklassens analysklasser (förenklat uttryckt funktionalitetsgrundande element). Figur 13 nedan illustrerar hur stereotyper för analysklasser kan kopplas till applikationen och dess funktioner. Ett exempel som kan påvisas är hur funktionaliteten Entity relateras med arkitekturella mekanismer för Persistence, Auditing och Error Management. Ansatserna är generella och inte speciellt detaljerade men Persistence kan här antas utgöra och tillföra arkitekturell funktionalitet till entiteten vad gäller dess lagring i systemet. Auditing kan sägas ha med entitetens revisionshantering att göra, hur entiteten förändras under sin livslängd (Persistence) i systemet. Error Management är en arkitekturell mekanism som tillför systemet funktionalitet för entitetens felhantering.



Figur 13. Illustrerar sambanden mellan systemfunktionalitet i form av stereotypa analysklasser och arkitekturella mekanismer (Svensson 2004-05-07).

7. Resultat

Den här rubriken ämnar presentera information härledda ur uppsatsens teoridel som direkt knyter an till besvarandet av uppsatsens frågeställningar. Resultatets resonemang och tankegångar återger utöver vad som kan härledas utifrån uppsatsens teoridel även material från intervjuer och vägledande samtal med metodexpert Sigvard Svensson. Vad som dokumenterats i samband med Intervjun finns som presenterat som bilaga med kompletterande att ljudupptagning att tillgå om så önskas.

Resultatet av litteraturstudiens omfång påvisar att systemarkitektur som begrepp och fenomen är mycket viktigt för systemutvecklingsprocessers resultat. Både kvalitet och hållbarhet av en mjukvaruprodukt är tillsammans med flera andra faktorer direkt beroende av om arkitekturen får den plats, tid och resurser inom utvecklingsprojektet som krävs för att infria de fördelar en god arkitektur för med sig eller inte. Omvänt, det vill säga, om inte arkitekturen ges det utrymme och de resurser som krävs kan de positiva fördelarna den annars skulle föra med sig bytas ut mot sina respektive negativa motsatser som betydande kvalitets och hållbarhetsbrister.
Ett explicit tillämpande av systemarkitekturellt tänkande har visat sig vara en mycket stor och betydande del av en ideal systemutvecklingsprocess vars mål eller fokus, utöver framställandet av en färdig produkt, ligger tungt på kvalitetsaspekter. Arbetet med framställandet av en systemarkitektur är ett omfattande ansvarsfullt område eftersom det färdiga systemets funktionalitet och beteende i stort är direkt beroende på den underliggande service systemarkitekturen erbjuder systemfunktionaliteten. Men med den arkitekturcentriska systemutvecklingssyn som RUP har låter det sig göras.

7.1. Huvudfråga

1. **På vilket sätt kan det tydligare framgå att ett färdigt projekt inte enbart överlämnar produktfunktionalitet till beställaren utan också levererar en systemarkitektur då det motiverar arkitekurella frågors relevans för beställaren?**

   Ur uppsatsens teoridel är följande det resultat som arbetats fram gällande uppsatsens huvudfråga:
   Arkitekturen måste låtas ta ännu större plats inom utvecklingsprojekten än tidigare och det låter sig enbart göras om man väcker en medvetenhet och sedermera ett genuint intresse hos kunden. Det är kunden som står för utvecklingskostnaderna där arkitekturen ingår. Medvetenheten och intresset i sin tur ska verka för att kunden i framtiden kräver att beställda system ska grunda sig på en enligt alla befintliga kriterier kvalitativ och genuin systemarkitektur. Den ökade medvetenheten väcks hos kunden genom att framhållandet av arkitekturens fördelar sker på ett än mer explicit och tydligt uttalat sätt än tidigare. Man måste visa på dess möjligheter genom ett aktivt krav- och analysförfarande enligt gängse arkitekturutvecklingsprinciper där betydelsen av aktiviteten betonas och lyftas fram så att en direkt koppling mellan arkitektur och det slutgiltiga systemets goda sidor blir synlig. Visar man tydligt på att

en god arkitektur är ett måste för ett högkvalitativt och i längden också hållbart system gör man det lättare för kunden att acceptera nyttan och nödvändigheten med att låta arkitekturen ta sin behövda plats. När det väl är i blickfånget blir det svårt att därefter bestrida arkitekturens välbehövliga plats i systemutvecklingsprojekt. Därmed kan man även visa på att det inte är kostnadsbesparande att fortsatt negligera framtagandet av en systemarkitektur utan snarare en bättre strategi att betala för att ge arkitekturverksamheten mer utrymme och resurser. Det krävs att kunden inser (får klargjort för sig) att arkitektur är en oumbärlig service för att systemets önskade funktionalitet och egenskaper ska infrias på bästa sätt ur ett strategiskt hållbart perspektiv.

I nuvarande läge verkar det som om huruvida vikten av arkitekturfrågor tas tillvara på tillbörligt sätt eller inte beror på arkitektens och projektledarens engagemang samt hur upplyst systembeställaren är om begreppet arkitektur och dess innebörder. Då det tillsynes saknas ett tydligt element i RUP med syfte om kundorienterad arkitekturell upplysning skulle en tänkbar lösning på problematiken vara att i processen infoga ett sådant element. Dess utförande skulle syfta till att marknadsföra vikten av arkitekturella frågor utöver den medvetenhet som redan finns idag.
I arbete med att väcka en ökad medvetenhet, ett genuint intresse och förståelse hos kunden är IEEEs FURPS+ modell för arkitekturell kravhantering tillsammans med Philippe Kruchtens 4+1-vymodell för systemarkitektur, rollen arkitekt och nyckelroller hos beställaren viktiga beståndsdelar.

Kontentan av utfallet angående uppsatsens huvudfråga är att inåt såväl som utåt jämställa systemarkiterell signifikans med klassisk systemfunkitonalitet genom ett kundorienterat arkitekturupplysande element.

7.2. Delfrågor

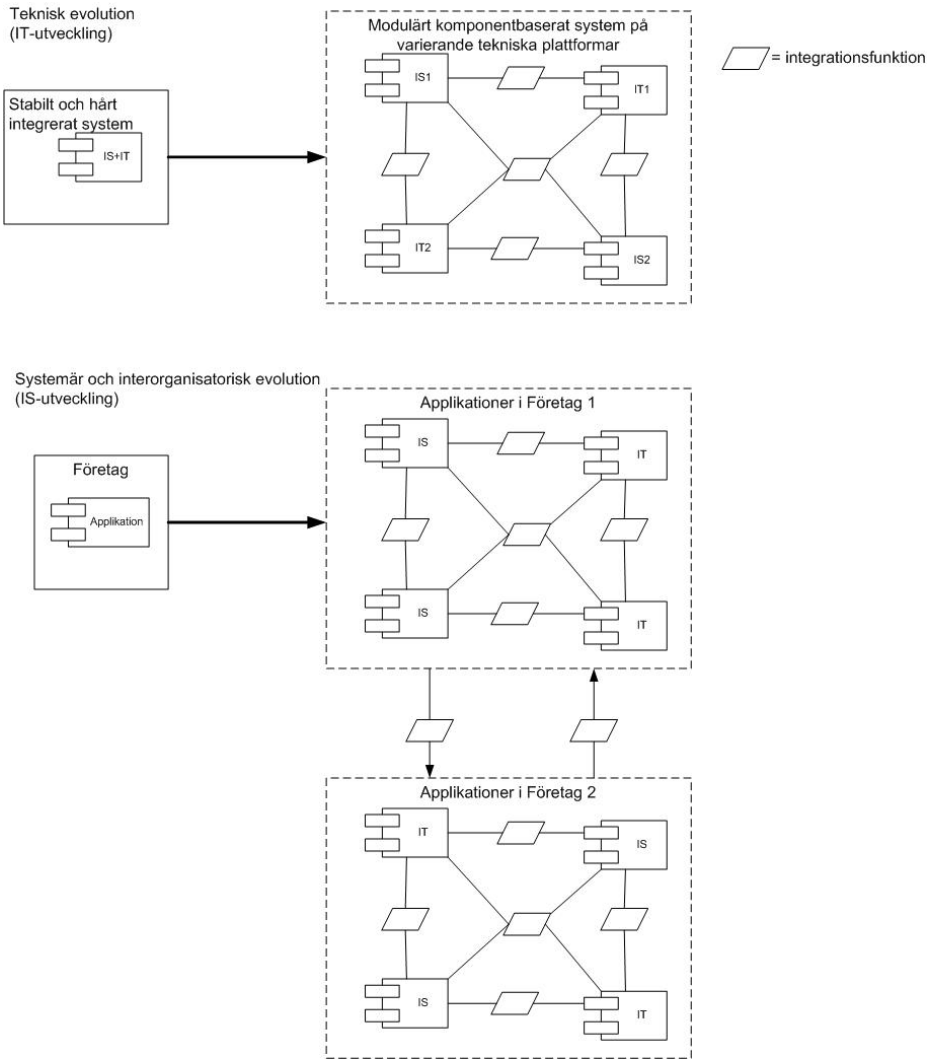
1. Varför är det viktigt att bli tydligare med arkitekturen samt vari ligger huvudfrågans bakgrund?

Det är viktigt att bli tydligare med arkitektur eftersom alla de fördelar den för med sig annars inte infinner sig och man förlorar bland annat kvalitet och hållbarhet hos produkten. Avsaknaden av de positiva konsekvenser en korrekt utformad arkitektur medför kan omvänt få vittgående negativa och kanske rentav ödesdigra konsekvenser för produktens kvalitet och livslängd. Betydelsen av det här har inte kommit till kundens insikt vilket gjort det svårt att få tillräckligt med resurser till arkitektonisk utveckling. Tidigare har mycket av det som arkitekturen förespråkar införlivats i produkter på ett mer implicit och outtalat sätt av designers med god insikt, erfarenhet och intuition som lyckats få med de nu mer tydligt definierade arkitektoniska kraven ändå. Idag sker detta på ett mer strukturerat sätt då det explicit finns utrymme för arkitektoniska frågor i utvecklingsprocessen. I väldigt många fall har dock de arkitektoniska fördelarna inte upptäckts eller inte tillåtits att ta plats vilket lett till

stora problem med just det som arkitekturer är tänkt att avhjälpa. Exempel är omfattande problematik som uppstår då ny teknik ska införlivas i systemen för att förlänga dess livslängd, höja prestanda eller dylikt.

Kontentan är att ju tydligare systemarkitekturen och dess positiva implikationer synliggörs för kunden desto bättre förutsättningar finns för att arkitekturfrågor får de resurser som behövs.

Bakgrunden till problematiken kan uttryckas som en evolution av arkitekturfokusering som skett både från tekniskt håll och systemeringshåll. Figuren 14 nedan syftar till att kortfattat illustrera hur en teknisk och systemmässig utveckling skett parallellt från hårt integrerade IT-system till mer komplexa modulära komponentbaserade system och från enskilda företagsapplikationer till interorganisatoriska samverkande informationssystem, från stabila till föränderliga miljöer.
Arkitekturbegreppet som disciplin eller aktivitet har utvecklats under de senaste 20 åren och fått ett ordentligt uppsving de senaste fem. Det kan sägas komma som ett naturligt resultat av att system genomgått en exponentiell ökning av såväl inneboende som omgärdande komplexitet vilket lett till att man successivt varit tvungen att frångå ursprungliga stabila system och (stordator)miljöer till föränderliga, öppna och komplexa system.
Ett behov av en explicit arkitekturdisciplin har växt fram och tillåtits ta plats vartefter insikterna om arkitekturens nödvändighet mognat.



Figur 14. Ska illustrera den tekniska och systemära evolutionen av arkitekturfokusering. Från hårt integrerade och stabila miljöer till tekniskt modulära, komponentbaserade och interorganisatoriska miljöer.

2. Vad är en systemarkitektur i allmänhet och i synnerhet i anknytning till RUP?

I allmänhet är en systemarkitektur den fundamentala organisationen av ett system och dess beståndsdelar som helhet. Aspekterna som tillsammans utgör arkitekturen inkluderar statiska såväl som dynamiska element, hur de samverkar sinsemellan inom systemet samt hur de utgör systemets beteende utifrån sett. Vidare inkluderas också systemets arkitekturella stil som visar de bakomliggande tankegångarna som lett fram till arkitekturens sammansättning. Arkitekturbegreppet involverar också andra egenskaper gällande systemets prestanda, skalbarhet, återanvändningsgrad, ekonomiska- och tekniska begränsningar.

Vad en systemarkitektur är inom RUP i synnerhet kan beskrivas i punktform men illustreras, synliggörs och betonas med hjälp av bland annat 4+1-vymodellen över arkitektur. Den är mycket viktig om man vill berätta vad en systemarkitektur medverkar till då vymodellen berättar, vad, av vilka delar, hur, när, för vem och varför om arkitekturen. Inom exempelvis arbetsflödet Architecture Analysis finns tydliga kopplingar till 4+1-vymodellen vilka kan påvisas för att ytterligare förstärka vikten av arkitektur och vymodellens samhörighet. Hos huvudartefakten för arkitekturfrågor där den slugiltiga artefakten, Architectural Document, för arkitekturen dokumenteras finns också tydliga kopplingar till vymodellens olika vyer. Vymodellen är alltså inte enbart en representation av systemets arkitektur utan kan även användas som verktyg i dess framställande.
I RUP definieras begreppet till att omfatta följande viktiga punkter där systemarkitekturen spelar en avgörande roll för slutprodukten:

- Helhetsförståelse av systemet
- Systemets funktionella och icke-funktionella egenskaper
- Systemets organisation
- Systemets struktur
- Systemets arkitekturella mönster
- Systemets arkitekturella stil
- Systemutvecklingsarbetets organisation
- Projektledning
- Graden av återanvändning
- Systemets vidareutveckling
- Systemets utbyggnadsprinciper
- Systemets inre och yttre samverkansprinciper

3. Vad gör systemarkitekturen betydande och vilka implikationer har den för systemutvecklingsprocessen?

Närmare presentation av ovan punkter sker i tidigare del av uppsatsen (se rubrik 4.3. Arkitekturens huvudsakliga syften) besvarar den här frågan på ett bra sätt då det är just betydelserna och implikationerna av punkterna som påvisar dess betydelse.

Tidigare omfattades de idag explicit uttryckta arkitekturbegreppen och deras innebörder av designers och rollen systemarkitekt fanns inte uttryckligen inom systemutvecklingen. Idag är det annorlunda då det finns bestämt utrymme och syfte för arkitekturverksamhet inom processen. Implikationerna som arkitekturen har fört med sig till utvecklingsprocessen grundas i att begreppet givits en klart uttalad och strategiskt betydande position inom processen. De positiva implikationer som arkitekturbegreppet har för den slutprodukt som utvecklingsprocessen resulterar i är många, allomfattande och av stor betydelse för resultatets totala kvalitet. Dess påverkan på processen och produkten finns tidigare redogjorda i punktform (se rubrik 4.3. Arkitekturens huvudsakliga syften) väl styrkta av teori och empiri inom området då en av RUPs grundläggande bästa praxis är arkitekturcentrisk utveckling.

Ytterligare en tydlig implikation på utvecklingsprocessen är en ökad grad av visuellt modellerande där införandet av standards inom området har spelat en viktig roll för att systemarkitekturen kunnat extraheras till en egen disciplin inom systemutvecklingen. Det är främst UML som ligger i åtanke när visuell modellering tas upp eftersom det är väl spritt, definierat och inte minst integrerat i RUP som utvecklingsprocess. Tidigare visualiseringsmetoder varierade mycket i kvalitet och dess semantiska ansatser var svåra att förstå för personer som inte direkt var involverade i arbetet med arkitektonisk utveckling. Logiken säger att det är mycket svårare att befästa någonting uttryckt på ett otydligt sätt än ett tydligt. En visuell modell uttryckt på ett etablerat bildspråk kan få stor genomslagskraft då dess innehåll förmedlas på ett sätt som är gemensamt. Framväxten av UML har således på ett tydligt sätt bidragit till arkitekturens uttryck och dess nuvarande plats inom systemutveckling. Ett viktigt exempel kan vara 4+1-vymodellen för arkitektur vilken både grundar sig på bland annat UML-modeller, dess egna visuella modelleringsmoment stöder sig på UML tillsammans med vymodellens egen presentation.

4. Hur arbetas en systemarkitektur fram inom ett RUP-baserat utvecklingsprojekt?

En systemarkitektur i ett systemutvecklingsprojekt arbetas fram gradvis och tar sin början i en arkitekturell kravhantering (FURPS+modellen) vilken syftar till att fånga arkitekturellt signifikanta krav, kartlägga och klassificera dem, finna lösande mekanismer till dem samt utvärdera och verifiera resultatet. Arbetet med arkitekturen

tar sin början i utvecklingsprojektets första förberedelsefas (Inception) i arbetsflödet för krav (Requirements) för att sedan övergå i fasen Etablering (Elaboration) och arbetsflödet för analys och design där bland annat arbetsflödesdetaljerna definiera kandidatarkitektur (Define a Candidate Architecture och förfina arkitekturen (Refine the Architecture). Detta tillsammans med exempelvis aktiviteterna arkitektonisk analys, användningsfalls (Use Case) analys i definiera kandidatarkitekturen och identifiera designmekanismer, identifiera deisgnelement med flera i förfina arkitekturen.
Med hjälp av FURPS+modellen erhålls en kravbild på arkitekturen som bland annat beskriver vilken service arkitekturen måste leverera till den övriga systemfunktionaliteten. Den arkitekturfunktionalitet och resultaten som härrör där ur utvärderas sedan gentemot de ursprungliga FURPS+kraven enligt definierade utvärderingskriterier för att kontrollera att kravbilden slutligen tillgodosetts. Resonemanger är att tillämpning av FURPS+modellen ger arkitekturen rötter som resulterar i en väl förankrad härkomst jämfört med en mer ad-hocbetonad arkitektonisk tillkomst. Man har helt annan kontroll över hela arkitekturen och det som den påverkar än vad man annars skulle ha och därmed ökar de positiva förutsättningarna för det totala systemet och dess uppbyggnad avsevärt.

Viktiga beståndsdelar i arkitekturarbetet som uppsatsen tagit upp är:

- **FURPS+modellen** för arkitekturell kravhantering.
- **Arbetsflöden** vilka utgörs av statiska processelement samlade för en disciplin med ett visst mål inom utvecklingsprocessen.
- **Arbetsflödesdetaljer** en förfinad logisk gruppering av RUPs statiska processelement utformad för att lösa mer specifikt uttalade problem inom processen.
- **Aktiviteter** enskilda moment som utförs av en specifik roll i projektet. Dess utförande resulterar i ett mätbart värde och de grupperas till att utgöra arbetsflödesdetaljer. Ytterligare ett statiskt processelement.
- **Iterationsplaner** här sker en anpassad tillämpning av statiska processelement. Detta utgör alltså i stort RUPs dynamiska processaspekter.

8. Diskussion

Diskussionsdelen i uppsatsen avser att göra tolkningar och dra slutsatser utifrån arbetets teoridel i relation till utfallet för frågorna som redovisats i rubriken resultat.

Valet att inte ta upp delfrågor 2 och 3 till diskussion har gjorts på grund av att en diskussion kring dem och deras utfall inte skulle gagna diskussionen kring huvudfrågans utfall. Delfrågor 2 och 3 är av sådan natur att grunden för deras utfall inte direkt förelägger någon diskussionsansats vars genomgång skulle tillföra något nytt utöver vad som redan presenterats i uppsatsens teori eller resultatdel. Att delfråga 1 tagits upp till diskussion före huvudfrågan motiveras av att en diskussion kring de båda frågeställningarna tillsammans och deras respektive utfall gör sig bättre på det viset.

8.1. Angående delfråga 1 och huvudfrågans bakgrund

Funderingar runt de ansatser till lösandet av uppsatsens frågeställningar leder osökt till tankar runt systemutvecklingens framväxt och mognad genom åren och en fråga som ställs är hur branschens mognad bäst avspeglas i kundernas värld? Budskapet som evolutionen av arkitekturfokus för fram har bevisligen gett resultat i form av explicit tilldelat utrymme inom systemutvecklingsprocessen RUP som till och med enligt dess grunddrag är arkitekturcentrisk. Är branschen dålig på att tillvarata chanser att utåt påvisa arkitekturens nödvändighet och att det är så att, även om det inom utvecklingsprocessen finns utrymme för arkitektur, det är i kundkontakten kommunikationen brister. Att den arkitekturella evolutionen ännu inte nått riktigt tillräckligt långt för att tillåtas väcka kundernas intresse. Eventuellt kan tänkas att systemarkitektur ur den oinvigdes perspektiv ter sig otydligt eller som någonting för tekniskt att engagera sig i. Ett annat resonemang som tycks relevant är att företagens kostnader för IS/IT-utveckling ses som något som bör minimeras så långt som möjligt istället för att inköpande organisationer ser sin IS/IT-utveckling som investeringar vilka måste ges en rimlig chans att låta betala sig. Informationssystem och informationsteknologi behandlar faktiskt en av organisationens viktigaste resurser – information. I ett sådant kortsiktigt kostnadsminimerande perspektiv där detta inte tillkännages kan tänkas att arkitektur är ett område som fått ge vika för mer omedelbart synlig systemfunktionalitet. De senare årens regression i branschen som bland annat lett till ett hårdnat konkurrensläge bland konsultbolag där exempelvis priser blivit ett viktigt konkurrensmedel har säkerligen inte verkat fördelaktigt för arkitekturfrågor i en prioriteringskamp om att sälja relativt snabbt tillgänglig funktionalitet framför en tillsynes dyrare sådan. Om än den senare med bättre kvalitets och hållbarhetsförutsättningar.

8.2. Angående uppsatsens huvudfråga

I uppsatsens teoridel finns en central rollfigur i arktektursammanhanget beskriven vilken är arkitekten. Arkitektens roll i arbetet med arkitektur är lättillgänglig på så sätt att den är väletablerad och har ett tydligt syfte tillsammans med tydliga föreställningar om hur rollens mål ska tillgodoses på bästa sätt. I jämförelse med detta lyser ett explicit orsak och verkanssamband för kundes arkitektoniska förståelse med sin frånvaro.

Resultatet pekar på att behovet av en modell med kunden, arkitekten och arkitekturen i fokus och det explicita målet att hjälpa arkitekten upplysa kunden om arkitekturens vikt skulle vara på sin plats. I ett nytt standardelement inom RUP skulle en sådan modell kunna stå i centrum för en kundorienterad workshop med syfte att tydliggöra just ett orsakssamband mellan arkitektonisk kravhantering samt utformning och realisering av systemfunktionalitet i form av realisering av användarfallsscenarier. På så sätt erhålls ett icke-exkluderande helhetsperspektiv där både arkitektur och systemfunktionalitet gemensamt utgör den centrala ståndpunkten. Modellens huvudsyfte skulle således vara att via den kundorienterade och förslagsvis arkitektledda workshopen belysa arkitekturens reella substans för den alltid så hett eftertraktade systemfunktionaliteten. Ett möjligt utrymme för detta aktivitetselement kan tänkas finnas i Inceptionfasen där grunden för projektets framtid till stor del avgörs. Avgörs det till exempel på det här stadiet att tillräckligt utrymme i form av tid och resurser inte kommer att tillhandahållas för arbete med arkitekturfrågor bör det klart och tydligt påvisa följdeffekter i projektomfattningen. Det ska synas att projektets genomförbarhet under tilltänkta omständigheter rimligen inte kan tillgodose alla sagda krav – kontentan ska visa att avkall på arkitektur leder till avkall på funktionalitet, kvalitet och hållbarhet och andra kvalitetsaspekter rörande systemet som helhet.
Då IT-branschen i dagsläget verkar gå mot ekonomiskt ljusare tider med ökad beläggning och orderingång till följd av att beställande organisationer känner positiva tendenser av en konjunkturuppgång tillsammans med eftersatt IT/IS-utveckling kan det finnas finansiella förutsättningar för ytterligare fokusering på arkitekturfrågor. Möjligheten för införandet av ett nytt arkitekturfokuserande element inom RUP kan således aktualiseras med bättre förutsättningar än tidigare.

8.3. Angående delfråga 4, hur en systemarkitektur arbetas fram

Förslag om lösandet av uppsatsens huvudsakliga problemställning vilken presenteras i arbetets huvudfråga skulle få direkta implikationer på dagens process med att arbeta fram en systemarkitektur. Införandet av ett nytt arkitekturfokuserande element inom RUP får naturligtvis konsekvenser för hur processen idag är uppbyggd kring arkitekturfrågor på så vis att en omarbetning av de processmoment som idag används förmodligen blir nödvändig. Hur omfattande en sådan omarbetning skulle kunna bli har författaren ingen bestämd uppfattning om men klart är att tillvägagångssättet vid en sådan omarbetning blir avgörande för vilken omfattning förändringen för med sig. Ett helt nytt moment för med sig tankar om en anpassning till den redan befintliga systemutvecklingsprocessen vilket om än för processen innebär ändrat utseende i sig inte behöver medföra allt för stora förändringar av dess befintliga delar. Medan en anpassning av den befintliga processen till att med befintliga medel inrymma en ny arkitekturfokuserande utåtriktad syn skulle tänkas medföra en mer omfattande omorganisation i processen. Förmodligen skulle resultatet bli en medelväg då det förvisso är frågan om ett nytt element med ett nytt syfte men inte något fundamental omarbete av processen i sig. Den är ju i sin utformning konstruerad på så vis att den låter sig anpassas efter omständigheterna och i övrigt uppdateras processen ständigt av dess upphovsmakare.

Förslag till vidare studier:
Som sig bör ska väl även lämnas förslag på fortsatta studier som helt eller delvis kan ha sin grund i denna uppsats och lämpliga områden att fortsätta arbeta med skulle kunna vara följande:

Kontrollera arbetets relevans till verkligheten genom att i empiriska undersökningar befästa eller förkasta dess teoretiska och akademiska ansats.

Om dess relevans befästs i empirin utarbeta en modell och ett tillvägagångssätt för hur uppsatsens föreslagna problemlösning bäst kan omsätts i praktiken.

Finns det relevanta kvantitativa ansatser som kan mäta eventuella framgångar eller motgångar i form av resursåtgång i förhållande till erhållet resultat av införandet av föreslagen lösning.

Angående arbetets aktualitet:
Det har inte varit möjligt att under arbetets gång arbeta med den senaste versionen av RUP som källa, åtminstone inte en fullständig version, utan enbart en tidigare version från 2001. Omarbete av delar i processen sker ständigt från en version till en annan där allt från arbetsflöden till terminologi ändras. Detta bör tas i beaktning vid eventuella framtida verksamheter av alla de slag där denna uppsats helt eller delvis ligger till grund. Den fortsatta bedömningen av uppsatsens relevans är dock att dess kärna fortfarande görs gällande i tillräckligt stor utsträckning för att uppfylla sitt sagda syfte.

9. Slutsats

Genom uppsatsarbetet har ett behov av en ny standardiserad beståndsdel i RUP med systemarkitekturellt upplysningssyfte med beställaren som målgrupp gjort sig gällande.

Slutsatsen pekar på att om inte arkitekturella frågor inom systemutvecklingen ges det utrymme i form av tid och resurser som krävs för att (på bästa sätt) infria de positiva egenskaper arkitekturen är ämnad att medföra leder det till risk att beställaren inte får det system han avsett. Risken för att ovan situation ska uppstå minimeras eller elimineras om beställaren görs varse arkitekturens betydelse för det levererade systemet på ett aktivt och explicit sätt. Orsakssamband mellan fångst, kartläggning, klassificering, lösning samt implementation av arkitekturellt signifikanta krav och den ”klassiska” systemfunktionaliteten ska visas på ett direkt sätt. Klarheten och insikten som då erhålls syftar till att påvisa hur medvetet eller omedvetet avkall på arkitekturella frågor explicit påverkar funktionaliteten, kvaliteten och hållbarheten hos det färdiga systemet. Viktigt är dock att en simplifiering av orsakssambanden in leder till en trivialisering som kan avfärdas utan att arktekturfrågornas integritet bibehålls. Inte heller ska arkitekturfokuserande och upplysande workshop eller underliggande modell vara för teknisk i sin ansats då den främst kan tänkas vända sig till och tilltala personer i icke-tekniska positioner. Målgruppen för arkitekturell fokusering hos beställaren bör identifieras som nyckelpersoner med inflytande över beslutsfattande angående projektets resurser.

11. Källförteckning

Böcker

Avison, E., D., Fitzgerald, G.: *Information Systems Development: Methodologies, Techniques and Tools 2nd Edition*, McGraw-Hill, 1995

Backman, J.: *Rapporter och uppsatser, Studentlitteratur*, Lund, 1998

Checkland, P., Scholes, J.: *Soft Systems Methodology: a 30 year retrospective*, Wiley, 1999

Clements, A.: *Principles of Computer Hardware*, Oxford University Press Inc., New York, 2000

Gustafson, D.: *Schaum's Outline of Software Engineering*, McGraw-Hill, New York, c2002

Kroll P., Kruchten P.: *Rational Unified Process Made Easy*, Adison Wesley, 2003

Kruchten, P.: *Rational Unified Process: en introduktion, svenska upplagan*, Adison Wesley, Boston, 2002

Lunell, H.: *Fyra rundor med RUP*, Studentlitteratur, Lund, 2003

Mathiassen, L.: *Objektorienterad analys och design*, Studentlitteratur, Lund, 2001

Magoulas, T., Pessi, K.: *Strategisk IT-Managemen*t, Institutionen för Informatik vid Göteborgs Universitet, 1998

Scott, K.: *Unified Process Explained*, Adison Wesley, Boston, 2001

Shaw, M., Garlan, D.: *Software Architecture Perspectives on an Emerging Discipline*, Prentice Hall, Upper Saddle River, New Jersey, 1996

Sommerville, I.: *Software Engineering 6th edition*, Adison Wesley, 2001

Thurén, T.: *Vetenskapsteori för nybörjare*, Liber, Stockholm, 1991

Wallén, G.: *Vetenskapsteori och forskningsmetodik*, Studentlitteratur, Lund, 1993, 1996

Papers

Eeles, P.: *Capturing Architectural Requirements*, The Rational Edge, e-zine for the Rational community Nov01, Rational Services Organization, UK, 2001

Kruchten, P.: *Architectural Blueprints—The "4+1" View Model of Software Architecture*, IEEE Software, 1995

Pettersson, S.: *Rational Unified Process, En modell för kvalitetsfrämjande systemutveckling*, Institutionen för Informatik vid Göteborgs Universitet, 1999

Leslee Probasco: *The Ten Essentials of RUP The Essence of an Effective Development Process*, Rational Software White paper, 2000

Övrig dokumenation

Rational Unified Process, Version 2001A.04.00, Copyright © 1987 – 2001, Rational Software Corporation, All Rights Reserved, Portions © Copyright IBM Corporation 1999-2001

Rational Unified Process Win Evaluation for Windows 2000, Windows NT, Windows XP (Rational Unified Process Evaluation Assembly V2003.06.00 for Windows 2000, NT, XP), http://www14.software.ibm.com/webapp/download/preconfig.jsp?id=2003-12-18+03%3A37%3A57.902091R

Vägledande samtal, intervjuer och handledning

Sigvard Svensson, Metodexpert, AcandoFrontec AB

Wera Tegner Johansson, Institutionen för Informatik, Handelshögskolan, Göteborgs Universitet

Mats Wessberg, Consolidate Sweden AB

Thanos Magoulas, Institutionen för Informatik, Handelshögskolan, Göteborgs Universitet

Bilaga

Öppen intervju med metodexpert Sigvard Svensson, AcandoFrontec AB, 2004-05-07

Intervjun hade som mål att klargöra ett flertal viktiga nyckelpunkter för uppsatsen rörande dess huvudsakliga syfte i relation till problem område, problemställning och förslag till lösandet av problemställningen.
Inför mötet skickades ett dokument som beskrev det uppdrag en uppsats egentligen är, dvs. vad som egentligen i slutänden skulle levereras vid inlämningsdagen. Vilka formella krav stod uppställda som akademiska egenskaper uppsatsen tvunget skulle innehålla.
De frågor som ställdes inför mötet och lade grunden för hela mötets innehåll var direkt kopplade till intervjuns mål och var följande:

Huvudfråga:

Q1. Hur blir man tydligare med att man inte enbart överlämnar funktionalitet till beställaren utan också en arkitektur?

Delfrågor:

Q2. Bakgrunden till problematiken, t ex. varför är det viktigt att bli tydligare med arkitekturen, vad har gjorts tidigare och hur gör man idag?
Q3. Vad är en systemarkitektur i allmänhet och i synnerhet inom RUP?
Q4. Vilka implikationer har arkitekturen för utvecklingsprocessen?
Q5. Hur arbetar man fram en systemarkitektur inom ett RUP-baserat projekt?

Nedan framgår de premisser vilka föranledde mötet i första hand:

De frågor som jag skulle behöva hjälp med är ju först och främst de där jag inte kunnat läsa mig till svaret och där är nog fråga 2 nyckeln eftersom den bör förklara varför huvudfrågan egentligen finns samt ge material till undersökning och resonemang till ett möjligt svar på fråga 1.
I svaren på fråga 2 innefattas också dagens dominerande lösning i frågorna ”varför är det viktigt att bli tydligare med arkitekturen?” och ”hur gör man idag?” som ger upphov till motiv och argument för en alternativ lösning. Väl så långt bör det även finnas visst underlag till resonemang kring hur en ny lösning skulle kunna se ut.

Sammanfattad diskussion:

Q1. Hur blir man tydligare med att man inte enbart överlämnar funktionalitet till beställaren utan också en arkitektur?

A1. Man måste låta arkitekturen ta ännu större plats inom utvecklingsprojekten än tidigare och det låter sig enbart göras om man väcker en medvetenhet och sedermera ett genuint intresse hos kunden. Det är i alla fall så att det är kunden som står för utvecklingskostnaderna där arkitekturen ingår. Medvetenheten och intresset i sin tur ska verka för att kunden i framtiden kräver att beställda system ska grunda sig på en enligt alla befintliga kriterier kvalitativ och genuin systemarkitektur. Den ökade medvetenheten väcks hos kunden på så sätt att man framhåller arkitekturens fördelar på ett mer explicit och tydligt sätt än tidigare. Man visar på dess möjligheter såväl som på dess nödvändigheter genom ett aktivt krav- och analysförfarande enligt gängse arkitekturutvecklingsprinciper där betydelsen av aktiviteten måste betonas och lyftas fram så att en direkt koppling mellan arkitektur och det slutgiltiga systemets goda sidor blir synliga. Visar man tydligt på att en god arkitektur är ett måste för ett högkvalitativt och i längden också hållbart system gör man det lätt för kunden att acceptera nyttan och nödvändigheten med att låta arkitekturen ta sin behövda plats. När det väl är i blickfånget blir det svårt att därefter bestrida arkitekturens välbehövliga plats i systemutvecklingsprojekt. Därmed kan man även visa på att det inte är kostnadsbesparande att fortsatt negligera framtagandet av en systemarkitektur utan snarare en bättre strategi att betala för ge arkitekturverksamheten mer tid.
Det krävs att kunden inser (får klargjort för sig) att arkitektur i högsta grad är en oumbärlig service för att systemets önskade funktionalitet och egenskaper ska infrias på bästa sätt ur ett strategiskt hållbart perspektiv.
I arbetet med att väcka en ökad medvetenhet, ett genuint intresse och förståelse hos kunden är nog IEEEs FURPS+ modell för att fånga arkitektoniskt viktiga krav tillsammans med Philippe Kruchtens 4+1-vymodell för systemarkitektur nyckelverktyg.

Q2. Bakgrunden till problematiken, t ex. varför är det viktigt att bli tydligare med arkitekturen, vad har gjorts tidigare och hur gör man idag?

A2. Det är viktigt att bli tydligare med arkitektur eftersom alla de fördelar den för med sig annars inte infinner sig och man förlorar kvalitet. Avsaknaden av de positiva konsekvenser en korrekt utformad arkitektur medför kan omvänt få vittgående negativa och kanske rentav ödesdigra konsekvenser för produkten. Betydelsen av det här har inte kommit till kundens insikt vilket gjort det svårt att få tillräckligt med resurser till arkitektonisk utveckling. Tidigare har mycket av det som arkitekturen förespråkar införlivats i produkter på ett mer implicit och outtalat sätt av designers med god insikt, erfarenhet och intuition som lyckats få med de nu mer tydligt definierade arkitektoniska kraven ändå. Idag sker detta på ett mer strukturerat sätt då det explicit finns utrymme för arkitektoniska frågor i utvecklingsprocessen. I väldigt många fall har dock de arkitektoniska fördelarna inte upptäckts vilket lett till stora problem med just det som arkitekturer är tänkt att avhjälpa, exempelvis omfattande problem då ny teknik ska in i systemen för att förlänga dess livslängd.
Bakgrunden till problematiken kan beskrivas i ordalag så som evolution av arkitekturfokusering vilken primärt skett från tekniskt och systemhåll.
(Se skiss)

Q3. Vad är en systemarkitektur i allmänhet och i synnerhet inom RUP?

A3. Vad en systemarkitektur är i allmänhet och inom RUP i synnerhet kan beskrivas i punktform men illustreras, synliggörs och betonas i 4+1-vymodellen över arkitektur. Den är oerhört viktig om man vill berätta vad en systemarkitektur medverkar till. Kom ihåg att den berättar, vad, av vilka delar, hur, när, för vem och varför. Inom Architecture Analysis finns tydliga kopplingar till 4+1-vymodellen vilka kan påvisas för att ytterligare förstärka vikten av arkitektur och vymodellen. Vymodellen är alltså långt ifrån enbart en representation av systemets arkitektur.

Q4. Vilka implikationer har arkitekturen för utvecklingsprocessen?

A4. Implikationerna som arkitekturen har fört med sig tillutvecklingsprocessen grundas helt i att begreppet givits en klart uttalad och strategiskt betydande position inom utvecklingen.
Tidigare omfattades de idag explicit uttryckta arkitekturbegreppen och deras innebörder av designers, rollen systemarkitekt fanns inte uttryckligen inom systemutvecklingen. Idag är det alltså annorlunda eftersom det faktiskt finns bestämt utrymme och syfte för arkitekturverksamhet.
De positiva implikationer som arkitekturbegreppet har för den slutprodukt som utvecklingsprocessen resulterar i är många, allomfattande och av stor betydelse för resultatets totala kvalitet. Dess påverkan på processen och produkten finns tidigare redogjorda i punktform (rubrikhänvisning) såväl som figurform, väl styrkta av teori och empiri inom området.

Arkitekturbegreppet som disciplin eller aktivitet har utvecklats under de senaste 20 åren och fått ett ordentligt uppsving de senaste fem. Det kan sägas komma som ett naturligt resultat av att system genomgått en exponentiell ökning av såväl inneboende som omgärdande komplexitet vilket lett till att man successivt varit tvungen att frångå ursprungliga stabila system och (stordator)miljöer till föränderliga, öppna och komplexa system. Evolutionen av arkitekturfokusering har skett på både ett rent tekniskt plan såväl som ett organisatoriskt systemeringsplan där man gått från hårda integrerade lösningar till mer komplexa icke-statiska lösningar som brukas i föränderliga miljöer.

Införandet av standards inom visuell modellering har också spelat en viktig roll för att systemarkitekturen ska kunna extraheras till en egen disciplin inom systemutvecklingen. Det är främst UML som ligger i åtanke när visuell modellering tas upp eftersom det är väl spritt, definierat och inte minst integrerat i RUP som utvecklingsprocess. Tidigare visualiseringsmetoder varierade mycket i kvalitet och dess semantiska ansatser var svåra att förstå för personer som inte direkt var involverade i arbetet med arkitektonisk utveckling. Logiken säger att det är mycket svårare att befästa någonting uttryckt på ett otydligt sätt än ett tydligt. En visuell modell uttryckt på ett etablerat bildspråk kan få stor genomslagskraft då dess innehåll förmedlas på ett sätt som är gemensamt. Framväxten av UML har således på ett tydligt sätt bidragit till arkitekturens uttryck och dess nuvarande

plats inom systemutveckling. Ett viktigt exempel kan vara 4+1-vymodellen för arkitektur vilken både grundar sig på bland annat UML-modeller, dess egna visuella modelleringsmoment stöder sig på UML tillsammans med vymodellens egen presentation.

Uppsatsen går inte närmare in på att utreda förhållande mellan UML-notationens uppkomst och arkitekturföreteelsen.

Q5. Hur arbetar man fram en systemarkitektur inom ett RUP-baserat projekt?

A5. Man arbetar fram en systemarkitektur med hjälp av IEEEs FURPS+modell som står för Funktionality, Usability, Reliability, Performance, Supportability och + står för Constraints som Design requirements, Implementation requirements, Interface requirements, Physical requirements. Man bildar med hjälp av modellen en kravbild på arkitekturen som bland annat beskriver vilken service den måste kunna leverera till den övriga systemfunktionaliteten (Se Sigvards ppt). Kravbilden ska skapas med kunden som aktiv deltagare, ju mer kunden inser dess vikt desto bättre. Ett systems funktionella krav utgör en ”resa” för sig och arkitekturen genomgår även den en ”resa” i samma tappning, det vill säga med utgångspunkt i en kravbild. Arkitekturens resa är nog så viktig om man inte vill göra funktionalitetens framkomst ogjord.
Den arkitekturfunktionalitet och resultaten som härrör där ur utvärderas sedan gentemot de ursprungliga FURPS+kraven enligt definierade utvärderingskriterier för att kontrollera att kravbilden slutligen tillgodosetts.
Arkitekturen framarbetas i huvudsak genom olika aktiviteter men primärt är det Architectural Analysis-aktiviteten som dominerar då dess mål bland annat är att definiera en kandidat arkitektur för systemet.
Väldigt viktigt är att fastslå en tydlig utgångspunkt för systemarkitekturens utveckling så att mest basala kriterierna för dess syfte och existens kan göras gällande för hela projektet. Den mest grundläggande frågan man måste ta ställning till är huruvida det aktuella projektet erfordrar att en helt ny arkitektur tas fram eller om det redan finns en befintlig arkitektur som kan bära upp projektets resultat samt vad som gäller i respektive fall.
Resonemanger är att tillämpning av FURPS+modellen ger arkitekturen rötter som resulterar i en väl förankrad härkomst jämfört med en mer ad-hocbetonad arkitektonisk tillkomst. Man har helt annan kontroll över hela arkitekturen och det som den påverkar än vad man annars skulle ha och därmed ökar de positiva förutsättningarna för det totala systemet och dess uppbyggnad avsevärt.