Programvaruutveckling för Stora System

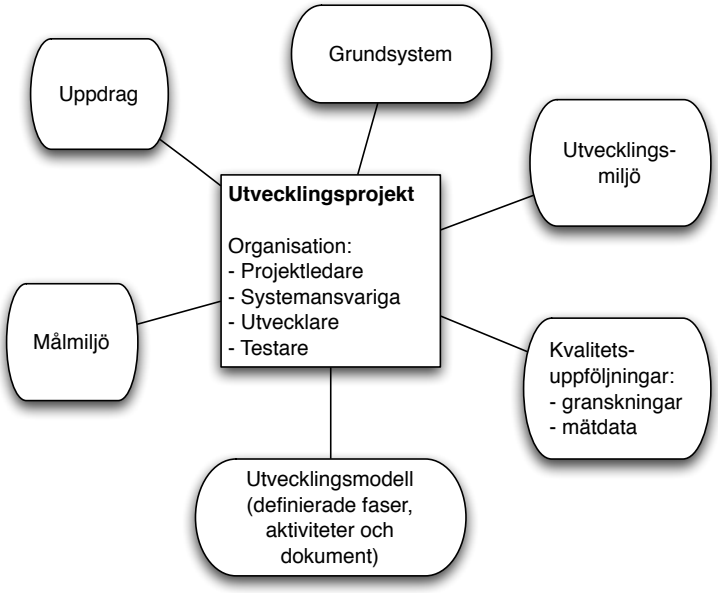
—

Projekthandledning

Institutionen för Datavetenskap
Lunds Tekniska Högskola, Lunds Universitet

Version 2.5

2018-09-06



Nummer	Aktivitet	Specificering
11	SDP	Arbete med dokument med bilagor.
12	SRS	
13	SVVS	
14	STLDD	
15	SVVI	
16	SDDD	
17	SVVR	
18	SSD	
19	Slutrapport	
21	Funktionstest	Arbete med testning
22	Systemtest	
23	Regressionstest	
30	Möte	Gruppmöte, expertmöte, etc
41	Föreläsning	Inlärning, "kurstid"
42	Övning	
43	Terminalövning	
44	Självstudier	
100	Övrigt	

Aktiviteter

U (Utveckling och dokumentation):
All den tid som läggs på att skapa saker från grunden.
I (Informell granskning):
Förberedelsetid och mötestid för informell granskning.
F (Formell granskning):
Förberedelsetid och mötestid för formell granskning.
O (Omarbete):
Den tid som går åt att göra förändringar.

Aktivitetstyper

Innehåll

Kapitel 1

Inledning

Programvarusystemen är några av de mest komplexa system som existerar, med extremt höga krav på tillförlitlighet, hårda realtidskrav, kontinuerlig utveckling av systemets programvara under systemets livstid, samt distribuerad realtidsmiljö.

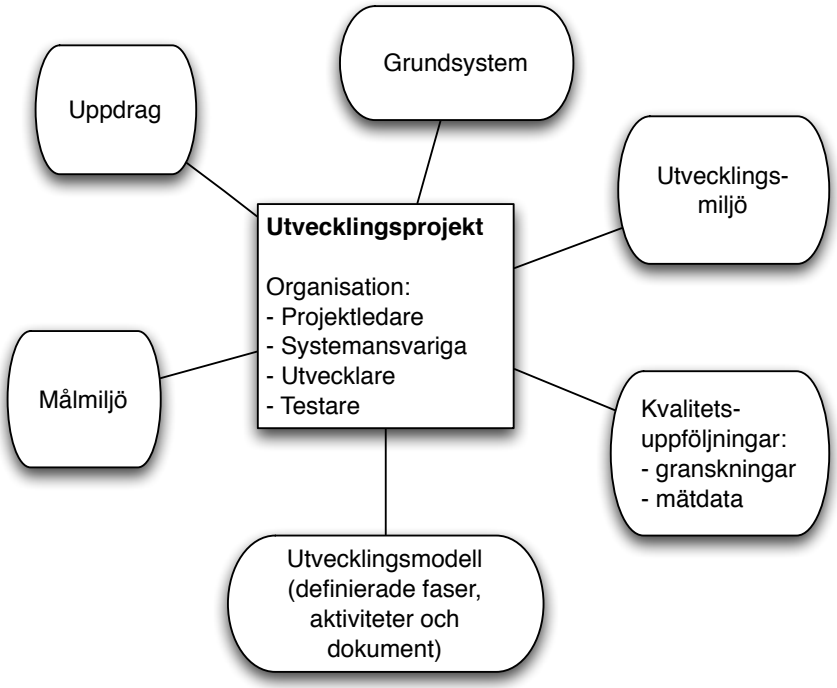
Programvara utvecklas fortfarande enligt en ganska primitiv metodik, samtidigt som kraven på slutprodukten hela tiden ökar. Kan vi t.ex. klara målet att utveckla ett system på mer än 100 000 rader programkod på en given tid, med en given kvalitet och till en given kostnad?

Nuvarande metoder leder förmodligen inte till att vi kan vara säkra på att vi lyckas. Vikten av en god metodik börjar bli allmänt erkänd och stora ansträngningar görs för att på ett effektivt sätt bygga in kvalitet i programvaruprodukter genom ett strukturerat och ingenjörsmässigt arbetssätt. Detta kompendium utgör ett stöd i kursen ”Programvaruutveckling för stora system”, och ger riktlinjer för hur ingenjörsmässigt utvecklingsarbete bedrivs i ett illustrativt programvaruprojekt. Projektets målsättning är att illustrera metoder och principer som behövs vid storskalig programvaruutveckling, samt att ge praktisk förståelse för centrala begrepp inom modern programvaruteknik.

1.1 Översikt av projektet

Kursen är planerad för 8 veckors (en läsperiod) deltidsstudier och omfattar 7,5 hp. Under dessa veckor ska en projektgrupp med definierade roller utveckla programvara enligt en initial uppgiftsbeskrivning. Arbetet kommer att bedrivas enligt en definierad utvecklingsmodell som beskriver en process som projektet ska arbeta efter. Denna process kommer att resultera i att ett antal definierade dokument produceras i enlighet med utvecklingsmodellen.

Det är mycket vanligare att ett programvaruprojekt berör vidareutveckling av ett befintligt system, än att ett helt nytt system utvecklas. Detta medför ofta speciella utmaningar, eftersom vidareutvecklingen måste ta hänsyn till det befintliga systemets designprinciper och historik. För att illustrera detta, skall projektet utgå ifrån ett *grundsystem* och vidareutveckla detta med ett antal nya tjänster. En del av arbetet kommer alltså att handla om att sätta sig in i grundsystemet och införa ändringar på ett sådant sätt att det passar väl in i grundsystemets struktur. I detta projekt utgörs grundsystemet av det system



Figur 1.1: Projekt och dess omgivning

som används på laborationerna, dvs Jetty, Jersey, H2 och Bootstrap.

Förutom den information som finns i detta kompendium behöver man även information om följande delar:

- Utvecklingsmiljön, som t ex Java och Eclipse om man programmerar i Java
- Grundsystemet, dvs det system som man vidareutvecklar i projektet.

Figur 1.1 visar projektet och dess omgivning. Ett uppdrag har givits av en kund, att utveckla ett antal nya funktioner till ett befintligt grundsystem för en specifik miljö. Som stöd för projektet finns en utvecklingsmiljö med språk och verktyg. Under projektets gång kommer kvalitetsuppföljning göras i form av granskningar, samt insamling och uppföljning av mätdata. Projektet arbetar enligt en utvecklingsmodell som definierar faser, aktiviteter och dokument. Projektet har en given organisation med olika roller för olika delgrupper (projektledare, systemansvariga, utvecklare och testare).

1.2 Grundläggande begrepp

Projekthandledningen är uppdelad i ett antal kapitel och appendix, och många av begreppen beskrivs och används i flera kapitel. Nedan presenteras en kortfattad beskrivning av de viktigaste begreppen, så att de olika kapitlen lättare

kan läsas oberoende av varandra. Utförligare förklaringar följer alltså i efterkommande kapitel och i de undervisningsmoment som ingår i kursen.

Vid en första genomläsning kan nedanstående begrepp verka många och svåra, men under projektets gång kommer de att ges en praktisk innebörd. Förklaringarna nedan är tänkta att utgöra en inledning, en referens och ett stöd för minnet.

Utvecklingsmodell (eng. process model) En utvecklingsmodell (även kallad processmodell) beskriver hur utvecklingen ska gå till och vad som ska produceras under utvecklingens gång. Utvecklingsmodellen föreskriver en process som utvecklingen ska följa. En process är det som sker när ett arbete utförs. En utvecklingsprocess för programvara utgör sålunda det som utförs under programutvecklingens gång. Processen innehåller ofta olika faser och aktiviteter som görs i en viss ordning, och resulterar i olika produkter (dokument och systemkomponenter).

Konfigurationshantering (eng. configuration management, CM) Under utvecklingens gång produceras en mängd dokument och systemkomponenter. Dessa växer ofta fram successivt och finns i många olika versioner. Dessutom behövs ofta olika versioner till olika kunder och plattformar. Ändringar behöver göras då fel eller problem upptäcks. För att hålla reda på ändringar och olika versioner behövs konfigurationshantering, vilken innefattar identifiering av konfigurationsenheter, ändringshantering och statusrapportering (se nedan).

Konfiguration (eng. configuration) Ett givet läge under eller efter utvecklingen, där alla produkter (systemkomponenter, dokument) har en given version.

Baseline Vid givna tillfällen under utvecklingens gång upprättar man s.k. baselines, som utgör en kontrollerad konfiguration, d.v.s. en väldefinierad samling av konfigurationsenheter som är enhetliga och passar väl ihop med varandra.

Konfigurationsenheter (eng. configuration item) En konfigurationsenhet är något som ska konfigurationshanteras, oftast är detta ett dokument, en del av ett dokument eller en systemkomponent.

Ändringshantering (eng. change management) Ändringshantering är en del av konfigurationshanteringen och innefattar rutiner för hur beslut om ändringar av systemet och dokumentationen tas och dokumenteras, samt hur information om ändringar sprids i projektet.

Statusrapportering Alla konfigurationsenheter har en status som anger dess version och vilka utestående ändringar som är under behandlande. Vid statusrapportering sammanställer man en versionshistorik för varje konfigurationsenhet.

Kravhantering (eng. requirements engineering) Ett programvarusystem byggs för att uppfylla ett antal krav. Kravhantering innefattar att på ett kontrollerat sätt samla in och specificera dessa krav, samt att säkerställa att de överensstämmer med uppdragsgivarens behov och önskemål.

Design Med utgångspunkt från kraven ges systemet ges en struktur som kan kallas arkitektur. Denna ligger till grund för systemets implementering.

Arkitektur (eng. architecture) Arkitekturen beskrivs systemets delar och hur de hänger samman. Ett annat ord för detta är högnivådesign.

Implementering (eng. implementation) Med utgångspunkt från systemets design, realiseras de olika systemkomponenterna i vad som kallas implementering. Man ser även ibland benämningen lågnivådesign, vilken då ofta föregås av en högnivådesign. Implementering är ofta detsamma som programmeringsfasen eller kodning.

Granskning (eng. review, inspection) Vid en granskning görs en strukturerad genomgång av ett dokument och ett möte hålls enligt vissa regler där beslut om åtgärd eller godkännande fattas, baserat på felen som hittats vid genomläsningen. I projektet finns två typer av granskningar: formella granskningar respektive informella granskningar. Vid de formella granskningarna deltar externa granskare, medan de informella granskningarna genomförs internt i utvecklingsprojektet.

Validering (eng. validation) När en produkt är färdig behöver den kontrolleras, så att man är säker på att den uppfyller kundens ursprungliga behov och önskemål. Vid validering kontrolleras att det är rätt produkt som utvecklats. Validering av systemet som utvecklats sker ofta genom granskning och ett s.k. acceptanstest som kunden utför.

Verifiering (eng. verification) För att säkerställa att en produkt uppfyller dess specifikation utförs en verifiering. Vid verifiering kontrolleras att produkten är korrekt utvecklad. Verifiering av systemkomponenter görs ofta genom testning, medan verifiering av dokument ofta sker genom granskning.

Testning (eng. test) Testning innebär att en systemkomponents exekvering kontrolleras. Vid testning exekveras systemkomponenten i en dator, och resultatet kontrolleras.

Kvalitetsutvärdering (eng. quality assessment) Medan verifiering och validering är inriktad på att kontrollera kvaliteten hos produkter, är kvalitetsutvärdering inriktad på att kontrollera kvaliteten hos processen. Kvalitetsutvärdering syftar till att säkerställa att processmodellen följs, och att processen är bra på att utveckla produkter med rätt kvalitet.

Mätningar (eng. metrics) Som stöd för projektledning och för utvärdering av processen och dess produkter behöver mätningar planeras, utföras, och sammanställas. Mätningar kan t.ex. gälla nedlagd tid på olika moment, storleken på olika systemkomponenter, produktiviteten för olika utvecklare, eller antalet upptäckta fel i olika produkter.

Projektbibliotek (eng. project library) Projektbiblioteket innehåller alla dokument och all kod som produceras under projektet.

Kapitel 2

Utvecklingsmodell

En utvecklingsmodell (eng. ”process model”) beskriver de steg man skall gå igenom och de dokument som skall produceras under utvecklingens gång. Utvecklingsmodellen föreskriver en process som utvecklingen ska följa. Processen innehåller olika faser och aktiviteter som görs i en viss ordning, och resulterar i olika väldefinierade produkter (dokument och systemkomponenter). Utvecklingsmodellen är också ett stöd i att kommunicera med omvärlden, t.ex. kunder eller marknadsavdelningen, om hur projektet fortskrider och är ett stöd för beslutsfattare i att ta beslut om t.ex. hur mycket pengar som ska satsas på ett visst projekt.

Detta kapitel beskriver den utvecklingsmodell som skall användas i projektet Vissa centrala aktiviteter och dokument i en industriell utvecklingsmodell har uteslutits ur projektet för att möjliggöra att projektet skall kunna genomföras inom den givna tiden. Vi har valt att inte stryka dessa aktiviteter och dokument helt ur beskrivningen nedan. Istället har vi lagt beskrivningar som gör att man inser att vissa aktiviteter och dokument har uteslutits. Detta gäller t.ex. användardokumentationen. Varje projekt har sina speciella förutsättningar och därför är det viktigt att anpassa generella modeller så att de passar det specifika projektet.

I detta kapitel kommer först en allmän översikt av modellen (kapitel 2.1), därefter kommer en uppräkning av vilka möten som ingår i projektgenomförandet (kapitel 2.2). Avsikten med detta är att mötena skall sättas i relation till utvecklingsmodellen. I samband med mötena anges också vem som förväntas deltaga i mötena. Det betyder att referens sker till roller i projektet som beskrivs i kapitel 3. Innan presentationen av detaljerna av utvecklingsmodellen beskrivs granskningar kortfattat (kapitel 2.3), då granskningar är en återkommande aktivitet i utvecklingsmodellen. Avsikten är att först ge en helhetsbild innan alla detaljer för en specifik fas presenteras.

2.1 Översikt av utvecklingsmodellen för projektet

Projektet skall följa en utvecklingsmodell som innehåller fyra faser. Målsättningen i detta avsnitt är att ge en allmän överblick och förståelse för helheten

innan de respektive delarna behandlas. Faserna överlappar varandra i tid, så ett projekt kan arbeta i flera parallella faser samtidigt.

Fas 1: Specifikation Här formuleras kraven i en kravspecifikation (SRS - Software Requirements Specification) och projektets planering dokumenteras i en utvecklingsplan (SDP - Software Development Plan). Planeringen för testning och kvalitetskontroll av dokument och systemkomponenter dokumenteras i verifierings- och valideringsspecifikationen (SVVS - Software Verification and Validation Specification). Dessa tre aktiviteter genomförs parallellt och avstämning mellan de olika aktiviteterna är mycket viktig, då det är i denna fas som basen för projektet läggs.

Fas 2: Högnivådesign och Testinstruktioner Här bestäms systemets struktur och hur systemets delar kommunicerar för att uppfylla kraven. Detta arbete dokumenteras i högnivådesignen (STLDD - Software Top Level Design Document). Vidare skriver man testinstruktioner baserat på de testfall som bestämts i SVVS. Instruktionerna för hur test skall genomföras dokumenteras i en testinstruktion (SVVI - Software Verification and Validation Instruction).

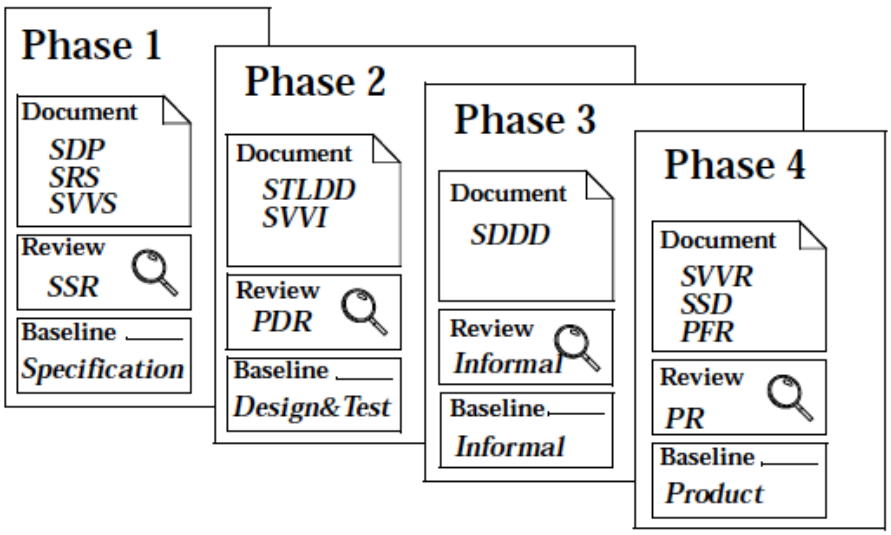
Fas 3: Lågnivådesign Här implementeras högnivådesignen och alla systemkomponenter specificeras fullständigt. Detta dokumenteras i lågnivådesignen (SDDD - Software Detailed Design Document).

Fas 4: Integration och systemtest Här sätts systemkomponenterna samman till ett system (integration) och systemet verifieras i funktions- och systemtest. Verifieringen och valideringen som skett under projektets gång dokumenteras i en verifierings- och valideringsrapport (SVVR - Software Verification and Validation Report). När systemet har erforderlig kvalitet kan det levereras till kund och leveransen dokumenteras i en systemspecifikation (SSD - System Specification Document). Vidare dokumenteras erfarenheterna från projektet i en slutrapport (PFR - Project Final Report).

Efter varje fas i utvecklingsmodellen utförs en granskning av dokumenten och en kvalitetsutvärdering av projektet, varefter en baseline upprättas och projektet får avsluta fasen. Inför granskningsmötena läser granskarna noga igenom dokumenten och försöker hitta fel och problem. På granskningsmötet sammanställer man vad granskarna hittat och efter att kommentarerna åtgärdats kan dokumenten ”frysas”. Detta kallas för baseline, vilket innebär att den fortsatta utvecklingen baseras på dokumenten i denna baseline och att ändringar i dessa måste ske på ett kontrollerat sätt.

Fas 1 avslutas med en krav- och testspecifikationsgranskning (SSR- Software Specification Review) och en Specification Baseline. Fas 2 avslutas med en designgranskning (PDR - Preliminary Design Review) och en Design and Test Baseline. Fas 3 avslutas med en projektintern (informell) granskning i form av ett statiskt test. Efter detta test går SDDD i baseline. Fas 4 avslutas med en produktgranskning (PR - Product Review) och en Product Baseline.

Figur 2.1 sammanfattar utvecklingsmodellen och vilka dokument, granskningar och baselines som ingår i respektive fas i utvecklingsmodellen.



Figur 2.1: Utvecklingsmodellen

2.2 Möten i utvecklingsprocessen

Följande typer av möten skall genomföras under projektgenomförandet, utöver granskningsmöten:

- Möten inom projektgruppen ska hållas 1–2 gånger varje vecka.
- Möten med sektionschef/kursansvarig ska hållas en gång i veckan. (Projektledargruppen) Dessa sker preliminärt på onsdagar lunchtid.
- Möte med kravexpert skall ske inför SSR, dvs inledningsvis i projeket. (Systemgruppen)
- Möte med testexpert skall ske inför PDR. (Testgruppen)
- Möte med designexpert skall ske inför PDR. (Systemgruppen)

Mötena beskrivs i detalj i kapitel 5.

2.3 Granskningar

Granskningar genomförs i alla faser i utvecklingsmodellen. En stor fördel med granskningar är att de kan genomföras på alla olika typer av dokument till skillnad från tester som bara kan genomföras på exekverbara produkter. Det medför att man kan använda granskningar tidigare i ett projekt, och därmed minska kostnaderna för att åtgärda eventuella fel. I detta avsnitt ges en kort introduktion till granskningar medan mer detaljer om granskningar beskrivs i kapitel 6.

Projektet skall bedriva informella och formella granskningar under fasernas fortskridande. De formella granskningarna är beskrivna i utvecklingsmodellen och de informella skall beskrivas i SDP

Projektet skall under granskningen dokumentera resultatet (information om fel som hittats) i ett s.k. granskningsprotokoll. Förändringar i senare faser som medför ändringar i de tidigare fasernas dokumentation får till följd att tidigare dokument skall uppdateras.

Om tidigare dokument är i baseline måste uppdateringen ske enligt definierad ändringhanteringsprocess, se kapitel 4.4.

Det finns även verktyg för att stödja granskningar av kod under ett projekts gång. Gerrit[1] är ett känt verktyg som stödjer att göra ändringar och tillägg i koden, bjuda in granskare att granska den ändrade koden, göra granskningar, samt att baserat på granskningsresultat (”+1”, ”+2”, etc) besluta om förändringen ska godkännas och läggas in i koden. Detta kompendium är dock skrivet utifrån de faser och dokument som normalt granskas ”på vanligt sätt”.

2.4 Anpassning

En given utvecklingsmodell skall i princip följas. Det är dock möjligt att man av olika skäl inte tycker att modellen fungerar eller att den på ett onaturligt sätt påverkar projektdeltagarna på ett sådant sätt att det har en negativ inverkan på projektet. Om så är fallet skall anpassningar av utvecklingsmodellen övervägas. Det är dock viktigt att ha följande i åtanke vid anpassning av utvecklingsmodeller:

- Varje aktivitet och dokument är introducerat av ett specifikt skäl, oftast för att hantera eller lösa ett specifikt problem.
- Innan man funderar på att ändra utvecklingsmodellen är man skyldig att identifiera vilket problem det är man försöker lösa.
- Då man identifierat problemet måste man fundera på alternativa sätt att hantera problemet. Om man inte kommer fram till något sådant bör man inte ändra utvecklingsmodellen.
- Det är således inte förrän man har en alternativ lösning på problemet som man har rätt att ändra i utvecklingsmodellen.

Om det är motiverat att genomföra en ändring av utvecklingsmodellen skall detta dokumenteras i SDP. Detta betyder att man i SDP skall ange vilka delar av utvecklingsprocessen man avser att använda samt noga dokumentera eventuella avvikelser. Det är viktigt att anpassningen motiveras i SDP och att kunden får granska och godkänna ändringen. Även anpassning av enskilda dokument är tillåten men vilka som ändras från sitt orginalformat skall anges i SDP.

[1] `http://www.gerritcodereview.com`

2.5 Fas 1: Specifikation

2.5.1 Inledning

I specifikationsfasen formuleras kraven på programvaran, testfall som skall utföras specificeras och projektplaneringen genomföras. Fasen avslutas med en formell granskning, Software Specification Review (SSR) och en formell baseline, Specification Baseline (SBL), upprättas.

2.5.2 Målsättning

Målsättningen med fasen är att fullständigt analysera, definiera och dokumentera kraven på programvaran, specificera vilka testfall som behövs för att testa systemet samt att ta fram en projektplan. Detta resulterar i dokumenten Software Development Plan (SDP), Software Requirement Specification (SRS) och Software Verification and Validation Specification (SVVS).

2.5.3 Aktiviteter

Software Development Plan (SDP)

Projektplanen (SDP) för projektet skall innehålla:

1. En utvecklingsplan som anger fasernas tidsbehov och händelser av betydelse (granskningar och baselines). I utvecklingsplanen beskrivs också eventuell anpassning av utvecklingsmodellen samt dokument till projektet (eng. tailoring).

2. En beskrivning av personalorganisationen som anger hur mycket personal som behövs, när de behövs, och var de behövs. Ansvarsförhållande och ansvarsområde för respektive person skall också anges.

3. En beskrivning av hjälpmedel, metoder och standarder som projektet avser att använda.

4. En beskrivning av konfigurationsstyrningen (CM), se kapitel 4. Konfigurationsstyrningen beskriver hur projektbiblioteket är organiserat och hur ändringshanteringen fungerar.

I samband med projektplanen är det också viktigt att tänka på att det i slutet av projektet skall skrivas en slutrapport. Detta betyder att man redan vid skrivandet av planen bör överväga hur slutrapporten skall se ut så att man enkelt kan stämma av med projektplanen.

Software Requirements Specification (SRS)

Alla krav på systemet måste identifieras, analyseras och dokumenteras. Detta samlas i Software Requirements Specification (SRS).

Software Verification and Validation Specification (SVVS)

SVVS är till för att planera testfasen och beskriva vad som skall göras, d.v.s vilka testfall som skall köras. SVVS är indata till SVVI, där man beskriver hur testen skall genomföras. Projektet skall analysera och bestämma sina metoder för:

Granskningar Då en granskning är genomförd skall granskningsprotokollet införas in i projektbiblioteket. I fas 4 införs alla de formella granskningarnas protokoll till Software Verification and Validation Report (SVVR). Man skall även ange vem som bestämmer om granskningen kan godkännas eller ej.

Funktionstest Testresultaten från funktionstesten skall införas i projektbiblioteket. Det är testarna som är ansvariga för att testa och godkänna en funktion. Vill man göra på något annat sätt skall det anges i SVVS. Testresultaten från testet skall införas i SVVR.

Systemtest Testresultaten från systemtestet skall införas i projektbiblioteket. Testresultaten från det formella systemtestet skall i fas 4 införas i SVVR. Det skall anges i SVVS vem som bestämmer om systemtestet kan godkännas eller ej.

Regressionstest I SVVS skall också tydligt anges hur regressionstest skall hanteras. Med regressionstest menas den kontinuerliga process som utförs för att säkerställa att tidigare korrekta delar fortfarande fungerar efter att ändringar eller uppdateringar har skett på systemet. Denna process skall beskrivas i SVVS.

I specifikationen skall även anges vad som skall testas i utvecklingsmiljön respektive på målmaskinen. SVVS avslutas med ett appendix där samtliga testfall skall beskrivas kortfattat.

2.5.4 Dokument som skall genereras under fasen

SDP, SRS, SVVS

2.5.5 Granskningar och baselines

Granskningar

Fasen avslutas med en formell granskning, Software Specification Review (SSR). Under granskningen skall projektet presentera dokumenten SDP, SRS och SVVS. I denna granskning ingår även genomgång av dokumentbibliotekets organisation och konfigurationsidentifieringslista. Syftet med granskningen är, bland annat, att visa att:

- SDP är en väl anpassad plan till projektet.
- SRS innehåller kundens alla krav på systemet.
- SVVS beskriver funktionstest, systemtest, regressionstest och granskningar på ett korrekt sätt.
- Projektet har god ordning på sin dokumentation.

Baselines

Fasen avslutas med att en formell baseline sätts, den s.k. Specification Baseline (SBL). När SDP, SSR och SVVS är klara och kunden har godkänt dokumenten kommer de att ingå i baseline. I baseline kommer också att ingå ändringar i dokumenten som är gjorda efter upprättandet av denna baseline. Efter det att dokumenten satts i baseline skall de hanteras enligt given ändringshanteringsprocess

2.6 Fas 2: Högnivådesign och testinstruktioner

2.6.1 Inledning

Under denna fas struktureras programvarusystemet i högnivåkomponenter och eventuella detaljer i kritiska delar detaljstuderas. Vidare skrivs testinstruktioner som anger hur de specificerade testfallen skall genomföras. Fasen avslutas med en formell granskning, Preliminary Design Review (PDR).

2.6.2 Målsättning

Målsättningen med fasen är att få fram en modulär och flexibel arkitektur på problemet i högnivåkomponenter, samt att specificera hur dessa högnivåkomponenter samverkar. Dessutom är målsättningen att formulera testinstruktioner som bland annat täcker samtliga krav.

2.6.3 Aktiviteter

Högnivådesign, Software Top Level Design Document (STLDD)

Projektet skall utifrån kraven i SRS dela upp systemet i högnivåkomponenter.

Testinstruktioner, Software Verification and Validation Instruction (SVVI)

Projektet skall utgående från testspecifikationerna i SVVS skriva testinstruktioner. Instruktionerna skall skrivas på ett sådant sätt att man skall kunna genomföra testen utgående från instruktionerna.

Eventuell lågnivådesign av kritisk programvara

Om det finns delar som är kritiska med avseende på säkerhet, prestanda, tillförlitlighet eller andra egenskaper, är det lämpligt att analysera dessa närmare i denna fas.

2.6.4 Dokument som skall genereras under fasen

STLDD, SVVI.

2.6.5 Granskningar och baselines

Granskningar

Fasen avslutas med en formell granskning, Preliminary Design Review (PDR). Under granskningen skall projektet presentera STLDD och SVVI, samt eventuella uppdaterade versioner av SRS och SVVS. Syftet med granskningen är att:

- Kontrollera överensstämmelsen med kundens krav, d.v.s efter eventuell uppdatering.
- Visa att högnivådesignen överensstämmer med SRS.
- Visa att testfall, som överensstämmer med kraven (SRS), är framtagna för att kunna verifiera och validera systemet.

Baselines

Fasen avslutas med en formell baseline, den s.k. Design and Test Baseline (DT-BL). När STLDD och SVVI är klara och kunden godkänner dokumenten kommer de att ingå i baseline. I baseline kommer också att ingå ändringar i dokumenten som är gjorda efter upprättandet av denna baseline. Efter baseline skall dokumenten hanteras enligt given ändringshanteringsprocess

2.7 Fas 3: Lågnivådesign

2.7.1 Inledning

Under denna fas struktureras programvarusystemet i enheter, som specificeras fullständigt. Enheterna skall implementera den av SRS:en krävda funktionaliteten. Då alla enheter är fullständigt specificerade implementeras (eller genereras) kod. Fasen avslutas med en informell granskning.

2.7.2 Målsättning

Målsättningen med fasen är att fullständigt beskriva alla enheter på ett förståeligt sätt, d.v.s konstruera en lågnivådesign som dels är testbar och dels underhållbar. Detta kommer att resultera i ett lågnivådesigndokument med programkod, Software Detailed Design Document (SDDD).

2.7.3 Aktiviteter

Projektet skall utifrån högnivådesignen utveckla programkoden, SDDD.

2.7.4 Dokument som skall genereras under fasen

SDDD

2.7.5 Granskningar och baselines

Granskningar

Efter den detaljerade designen utförs en informell granskning. Det är projektgruppens ansvar att tillse att SDDD implementerar det av kunden specificerade systemet, samt att dokumentet har en tillräckligt hög kvalitet för att kunna överlämnas för funktions- och systemtest. Syftet med granskningen är att övertyga sig om att den detaljerade designen uppfyller följande:

- Designen (STLDD och SDDD) överensstämmer med kraven i SRS.
- SDDD förfinar designdetaljerna från STLDD.
- SDDD överensstämmer med STLDD.

Baselines

Fasen avslutas med en informell baseline. Low-level Design Baseline upprättas då den informella granskningen är genomförd och eventuella fel från denna är korrigerade och godkända.

2.8 Fas 4: Integration och systemtest

2.8.1 Inledning

Under denna fas testas systemets funktionalitet. Detta sker först genom att funktionstest genomförs. Systemet integreras sedan, d.v.s de olika delarna sätts samman och ett systemtest genomförs.

Programvarusystemet testas och modifieras tills projektet försäkrat sig om att programvarusystemet uppfyller de specificerade kraven. Då systemet är godkänt i samtliga funktions- och systemtestfall ska slutdokumentationen produceras och systemet levereras till kund.

Fasen avslutas med en formell granskning, Product Review (PR) och en formell baseline, Product Baseline (PBL).

2.8.2 Målsättning

Målsättningen med fasen är att ha ett helt fungerande system i enlighet med de tidigare specificerade kraven.

2.8.3 Aktiviteter

Funktionstest

Projektet skall utföra funktionstest. Under arbetet med funktionstest skall projektet tillse att funktionstest genomförs i enlighet med SVVS och instruktionerna i SVVI.

Samtliga framtagna dokument som berörs skall uppdateras om fel uppkommer i test, d.v.s all levererad dokumentation skall överensstämma vid leveranstidpunkten. Utfallet från test skall dokumenteras i testrapporten (SVVR), samt att eventuella fel skall dokumenteras i avsedda problemrapporter.

Systemtest

Projektet skall utifrån kraven och planerna i SRS och SVVS testa systemet. Testfallen genomförs i enlighet med testinstruktionerna i SVVI. Systemtestet bör om möjligt genomföras av personer som ej varit inblandade i framställningen av systemet. Resultatet från det formella systemtestet resulterar i en testrapport som ingår i SVVR. Testrapporten skall innehålla:

- Sammanfattning av systemtestet.
- Detaljerad testhistoria.
- Testresultat.
- Utvärdering av testresultat och rekommendationer för eventuella ändringar.

Software Verification and Validation Report (SVVR)

Projektet skall framställa Software Verification and Validation Report (SVVR)

System Specification Document (SSD).

Projektet skall i System Specification Document (SSD) identifiera och beskriva det levererade systemet.

Project Final Report (PFR)

Efter genomförandet av projektet skall erfarenheterna dokumenteras i en slutrapport. I rapporten skall erfarenheter och mätetal från projektet dokumenteras. Det är speciellt viktigt att försöka generalisera sina erfarenheter, d.v.s skriva dem på ett sådant sätt att informationen kan vara av intresse för kommande projekt. Målsättningen med en slutrapport är att dokumentera erfarenheterna så att andra kan tillgodogöra sig dem. Det gäller att identifiera såväl positiva som negativa saker med projektet samt att försöka identifiera anledningarna till dessa och analysera hur de skulle kunna förstärkas respektive förhindras.

2.8.4 Dokument som skall genereras under fasen

SVVR, SSD, PFR

2.8.5 Granskningar och baselines

Granskningar

Fasen avslutas med en formell granskning, Product Review. Under granskningen skall tillverkaren presentera SVVR, SSD och eventuell användardokumentation samt all övrig dokumentation som behövs för att visa att programvarusystemet och dokumentation är komplett och att alla dokument överensstämmer sinsemellan.

Baselines

Fasen avslutas med en formell baseline, Product Baseline (PBL).

2.8.6 Efter leverans

Kunden genomför efter leverans ett acceptanstest av systemet. Vidare går kunden genom den levererade dokumentationen från projektet. Dokumentationen, utfallet från acceptanstest och intryck från projektet under dess genomförande ligger till grund för kundens uppfattning om projektet.

Kapitel 3

Personalorganisation

Utvecklingsprojektet utförs av en utvecklingsorganisation på uppdrag av en beställarorganisation.

I respektive organisation finns olika roller, vilket illustreras i figur 3.1. Utvecklingsorganisationen bildar ett projekt med en projektorganisation för att genomföra uppdraget. Följande deltagare finns definierade i projektet: Projektledargrupp (PG), Systemgrupp (SG), Utvecklingsgrupp (UG) och Testgrupp (TG). Följande roller finns utanför projektet: Sektionschef, Granskare, ett antal experter, samt en kund utanför utvecklingsorganisationen. Detta kapitel beskriver de olika rollerna och deras ansvarsområden och arbetsuppgifter.

3.1 Sektionschef

Sektionschefen är projektgruppens högste chef och ska hjälpa projektgruppen med eventuella icke-tekniska problem som kan dyka upp under projektets gång. Dock ska alla problem i projektet i görligaste mån först tas upp med projektledargruppen, som står för kontakten med sektionschefen.

Kursansvarig är sektionschef under detta projekt.

3.2 Kund

Kunden är den som ger projektgruppen dess uppdrag och är mottagare av resultatet. Kunden ska även godkänna den levererade produkten.

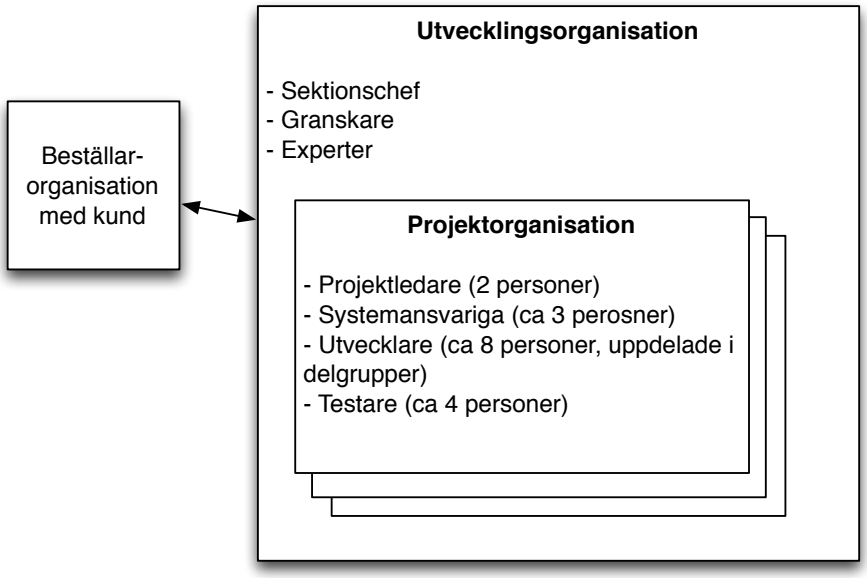
Kunden har rätt att under projektets gång kunna gå in i projektet och göra en extern kvalitetsutvärdering. Denna innefattar kontrollera att projektet följer processen och att status gentemot den framtagna utvecklingsplanen stämmer.

Kursansvarig är kund under detta projekt.

3.3 Granskare

Granskaren ska genomföra de formella granskningarna och kontrollera att utvecklingsmodellen följs. Utvärderaren ska under projektets gång, när som helst, kunna gå in i projektet och kontrollera dess status.

Lärare från institutionen är granskare under detta projekt.



Figur 3.1: Projektorganisation

3.4 Experter

Det finns ett antal experter att tillgå under projektets gång. Dessa experter kan rådfrågas angående frågor inom deras respektive expertis. Följande experter finns:

Kravexpert Hjälp vid framtagning av SRS.

Testexpert Hjälp vid framtagning av SVVS, SVVI och SVVR.

Designexpert Hjälp vid framtagning av STLDD och SDDD.

Dessa experter kan träffas efter överenskommelse. Vid problem med SDP, SSD och PFR ska sektionschef rådfrågas. Lärare från institutionen agerar experter under detta projekt.

3.5 Projektledargrupp

Projektledargruppen (PG) består av två personer vilka har det övergripande ansvaret för hela projektgruppen. PG:s huvudsakliga uppgifter är:

1. Ansvara för att gruppen presenterar ett resultat. Detta innebär att PG ska säkerställa att projektgruppen genomför arbetet, t ex genom att delegera det tekniska arbetet till de andra grupperna.

2. Bemanna projektgruppen och fördela arbetsuppgifterna. I samband med bemanningen är det viktigt att alla roller fylls, t ex ska en systemledare och en testledare utpekas.

3. Göra en detaljerad tidplan för projektet, i samråd med övriga projektmedlemmar, där man kan se vad respektive grupp ska göra dag för dag. Denna tidplan ska ingå i Software Development Plan (SDP).
4. Producera SDP och regelbundet stämma av projektplanen mot utfall. Inför PDR ska SDP:n uppdateras samt i övrigt vid behov.
5. Tidigt planera för att kunna producera en bra slutrapport efter projektet.
6. Göra en lista, i samråd med övriga projektmedlemmar, som identifierar samtliga konfigurationsenheter i projektet. Denna gås sedan igenom med sektionschefen. Projektledarna har också ansvar för att listan hålls uppdaterad och att eventuella tillägg och förändringar kommuniceras med sektionschefen.
7. Ansvara för att alla gruppmedlemmar får nödvändig utbildning på utvecklingsverktyg som används i projektet, konfigurationshantering, granskning, testning, etc.
8. Har rätt att omfördela arbetet inom gruppen om så krävs. Detta kan ske för att uppnå att tidplanen skall hållas och vidare att arbetsfördelningen blir jämn.
9. Sköta kontakten med kund, granskare och sektionschef.
10. Sammankalla till projektmöten och agera ordförande och sekreterare under dessa (1-2 gånger per vecka).
11. Fördela arbetet inför (informella) granskningar, speciellt ska inom projektgruppen utses ansvariga granskare för alla dokument.
12. Agera författare och granskare på informella granskningar
13. Agera författare på formella granskningar
14. Agera sekreterare under de formella granskningarna.
15. Deltaga i möten med sektionschef (1 gång per vecka).
16. Producera System Specification Document (SSD).
17. Ansvara för, samt löpande kontrollera att samtliga dokument finns i projektbiblioteket och att dessa är aktuella.
18. Ansvara för dokumentbiblioteket så att detta alltid finns tillgängligt för kund och granskare.
19. Deltaga i förändringskontrollgruppen tillsammans med systemgruppen.
20. Samla in (kontinuerligt) och sammanställa alla delgruppers metricsuppgifter.
21. Ta fram slutsatser och kommentarer till slutrapporten, rörande de moment som PG har varit med om.
22. Sammanställa slutrapporten samt metrics med slutsatser och kommentarer för hela projektgruppen.
23. Tidrapportera egen arbetstid.

3.6 Systemgrupp

Systemgruppen (SG) består normalt av 4 personer och har ansvar för det tekniska arbetet. En av gruppens medlemmar skall agera systemledare och ansvara för arbetsfördelningen inom gruppen samt rapportering till PG. Systemledaren är också ansvarig för samordning med testledaren vad avser konsistens mellan kravspecifikation (SRS) och testspecifikation (SVVS). De som inte är systemledare är ansvariga för kontakten med var sin utvecklingsgrupp. SG:s huvudsakliga arbetsuppgifter är:

1. Ha ett övergripande teknik- och systemansvar på delegation från projektledarna.
2. Utse systemgruppsledare, vars roll det är att se till att arbetet inom gruppen fördelas på bästa sätt.
3. Gemensamt med testgruppen ansvara för att SRS och SVVS är samstämmiga. Detta ska ske genom god kommunikation mellan systemledare och testledare samt granskning.
4. Arbeta tillsammans med utvecklingsgrupperna och deltaga i utvecklingsarbete. Ett särskilt ansvar ligger på systemgruppen att identifiera frågor som måste samordnas med andra utvecklingsgrupper, samt deltaga i denna samordning.
5. Agera författare och granskare på informella granskningar.
6. Agera författare på formella granskningar.
7. Sköta kontakten mellan utvecklingsgrupperna och testgrupperna. Avsikten med detta är att samordna så att dokument från UG och TG stämmer överens och är konsekventa.
8. Ansvara för uppbyggnad och underhåll av arbetsbiblioteket. Detta innebär ett ansvar för konfigurationshanteringen i projektet.
9. Ansvara för sammanställningen av de olika delgruppernas arbetsresultat till följande gemensamma dokument för hela projektet: SRS, STLDD och SDDD.
10. Ansvara för att de olika delgruppernas arbetsinsats, rörande de gemensamma dokumenten, är större än SG:s. SG måste alltså delegera arbete till UG.
11. Ansvara för och administrera fel- och ändringshanteringen samt statusrapporteringen för detta.
12. Deltaga i förändringskontrollgruppen tillsammans med projektledargruppen.
13. Tidrapportera egen arbetstid.
14. Sammanställa slutsatser och kommentarer till slutrapporten, rörande de moment som SG har varit med om.

3.7 Utvecklingsgrupp

Utvecklingsgruppen (UG) har hand om utvecklingen av funktionaliteten i projektet. Exempel på funktioner för utvecklingsgrupper är ”backend”, ”frontend” och ”algoritm”. Gruppen ska delas upp i delgrupper om två personer som har hand om utvecklingen av en funktionalitet vara. Delgrupperna kan t ex ha hand om områden som tidrapportering, sammanställning av mätdata, administration och databas i utvecklingen av ett tidrapporteringssystem, eller debitering, underhåll, vidarekoppling, vidarekoppling vid ej svar och samtalshämtning vid utveckling av ett telefonsystem. UG:s huvudsakliga arbetsuppgifter är:

1. Producera delkapitel för sin funktionalitet i följande dokument: SRS, STLDD, SDDD.
2. Utveckla funktionalitet enligt kravspecifikationen.
3. Vid behov assistera SG med ändringar i ett eventuellt grundsystem.
4. Agera författare och granskare på informella granskningar
5. Agera författare på formella granskningar.
6. Tidrapportera egen arbetstid.
7. Sammanställa slutsatser och kommentarer till slutrapporten, rörande de moment som UG har varit med om.

3.8 Testgrupp

Testgruppen (TG) har ansvaret för testningen av det utvecklade systemet. En av gruppens medlemmar skall agera testledare och ansvara för arbetsfördelningen inom gruppen samt rapportering till PG. Vidare skall testledaren tillsammans med systemledaren ansvara för konsistensen mellan kravspecifikation (SRS) och testspecifikationen (SVVS) TG:s huvudsakliga arbetsuppgifter är:

1. Producera de dokument som har med testning att göra: SVVS, SVVI och SVVR.
2. TG ska gemensamt med systemgruppen (i första hand systemledare) ansvara för att SRS och SVVS är samstämmiga. Detta ska ske genom god kommunikation mellan systemledare och testledare samt granskning.
3. Utföra och dokumentera samtliga tester enligt SVVS och SVVI.
4. deltaga i utvecklingsgruppernas arbete, t ex när det gäller att ta fram automatiska enhetstest.
5. Rapportera alla upptäckta fel under systemtest till SG.
6. Agera författare och granskare på informella granskningar.
7. Ansvara för och leda de informella granskningarna.
8. Agera författare på formella granskningar.

9. Ansvara för systembyggande.
10. Ansvara för att systemet regressionstestas.
11. Tidrapportera egen arbetstid.
12. Sammanställa slutsatser och kommentarer till slutrapporten, rörande de moment som TG har varit med om.

Kapitel 4

Konfigurationshantering

Under utvecklingens gång produceras en mängd dokument och systemkomponenter, som växer fram successivt och därför finns i många olika versioner. Dessutom behövs kanske olika versioner till olika kunder och plattformar. Ändringar behöver göras då fel eller problem upptäcks. För att hålla reda på ändringar och olika versioner behövs *konfigurationshantering* (eng. configuration management, CM). Konfigurationshantering innefattar bland annat identifiering av konfigurationsenheter, benämning av versioner, ändringshantering och statusrapportering. Det finns verktyg för att stödja versions- och ändringshantering, t.ex. SVN, men arbetet kan också göras manuellt.

För att många människor ska kunna arbeta på dokument och systemkomponenter parallellt behöver man koordinera detta arbete, så att man inte förstör för varandra. Detta kan åstadkommas genom att ge systemet och dess dokumentation en sådan struktur att det finns välavgränsade enheter sam kan hanteras var för sig. En sådan enhet, som man vill ska kunna ha en versionshistorik, kallas konfigurationsenhet. En konfigurationsenhet är alltså något som ska konfigurationshanteras, vilket oftast är ett dokument, en del av dokument, eller en systemkomponent.

En konfiguration (eng. configuration) är ett givet läge under eller efter utvecklingen, där alla systemkomponenter och dokument (konfigurationsenheter) har en given version. Med *systembyggande* menas att sätta ihop systemkomponenter till en exekverbar konfiguration.

Vid givna tillfällen under utvecklingens gång är det vanligt att man upprättar s.k. *baselines*, som utgör en kontrollerad konfiguration, d.v.s. en väldefinierad samling av konfigurationsenheter som är enhetliga och konsekventa (passar ihop). Innan man upprättar en baseline måste de ingående konfigurationsenheterna sättas samman och kontrolleras så att de är enhetliga och konsekventa.

Ändringshantering (eng. change management) är en del av konfigurationshanteringen och innefattar rutiner för hur ändringar beslutas och dokumenteras, samt hur information om ändringar sprids i projektet.

Alla konfigurationsenheter har en status som anger dess version och vilka utestående ändringar som är under behandling. Vid *statusrapportering* (eng. configuration status accounting) sammanställer man en versionshistorik för varje konfigurationsenhet.

Det är viktigt att varje projekt definierar hur konfigurationshanteringen skall gå till. Detta kapitel definierar rutiner för konfigurationshantering i utvecklings-

projektet och beskriver hur förändringar i baselines ska införas, hur dokument ska benämnas, samt hur status för införandet av förändringar dokumenteras.

4.1 Ansvarsfördelning

I utvecklingsprojektet är förändringskontrollgruppen (FKG) (eng. change control board) ansvarig för konfigurationshanteringen. FKG består av projektledargruppen och systemgruppen (FKG=PG+SG). Det är systemgruppen som har huvudansvaret, men projektledarna är med för att kunna fatta beslut om ändringsåtgärder som kräver resurs- och tidsplanering.

Identifiering, statusrapportering, och ändringshantering sköts av FKG för alla konfigurationsenheter och dessa aktiviteter startar omedelbart när projektet sätts igång.

Testgruppen är ansvarig för systembyggande och att delge denna kunskap till projektledargruppen, då denna producerar SSD i samband med systemleverans.

4.2 Identifiering av konfigurationsenheter

Redan första veckan skall projektet definiera vilka konfigurationsenheter och vilka formella dokument som ska produceras. Vidare definieras ansvarig för produktion av alla konfigurationsenheter och formella dokument. Detta resulterar i en lista som gås igenom tillsammans med sektionschefen. Listan benämns *konfigurationsenhetslista* och syftet med listan är att redan från början ha en god bild av vilka dokument och enheter som skall ha ett specifikt nummer, t.ex. dokumentnummer (se nedan), och därmed versionshanteras.

Under projektets gång kan det uppstå behov att identifiera ytterligare konfigurationsenheter, och i sådana fall skall sektionschefen informeras.

Konfigurationsenheterna numreras enligt följande syntax:

- först kommer en kort bokstavskombination som anger kontext det är frågan om (i denna kurs används ”PUSS”[1]),
- sedan kommer två siffror som anger vilket år dokumentet skapades,
- sedan kommer en siffra som anger vilken typ av dokument det är frågan om,
  - en nolla betyder att dokumentet ingår i kursmaterialet,
  - en fyra betyder att det är producerat av en projektgrupp,
- sedan kommer en siffra som anger vilken projektgrupp som dokumentet tillhör. Denna siffra finns ej med i dokumentnumren för kursmaterialet.
- sedan kommer två siffror som är löpnummer för projektets olika dokument.

Ett exempel på ett dokumentnummer är:

```
PUSS124819
```

[1] i vissa äldre dokument används ”MD”

vilket betyder:

`PUSS`	Dokument producerat i kursen Programvaruutveckling för Stora System
`12`	år 2012
`4`	producerat av projektgrupp
`8`	projektgrupp 8
`19`	löpnummer 19

4.3 Benämning av versioner

Numrering av versioner av ett system, en fil, ett dokument och andra delar av projektet ska göras enligt följande:

Första versionen kallas 0.1 och efterföljande 0.2, 0.3,..., 0.10, 0.11, o.s.v. Då första baseline upprättats ges konfigurationsenheten versionen 1.0. Efterföljande versioner kommer sedan att heta 1.1, 1.2 och så vidare. En baseline som släpps till kund benämns ofta *release*.

4.4 Ändringshantering

Ändringshantering är processen där en förändring av en konfigurationsenhet behandlas. En ändring föreslås, utvärderas, godkänns eller avslås, schemaläggs, åtgärdasoch slutligen kontrolleras den genomförda ändringen. Ändringshanteringen ska säkerställa att förändringar införs på ett kontrollerat vis, så att deras påverkan på systemet kan förutsägas.

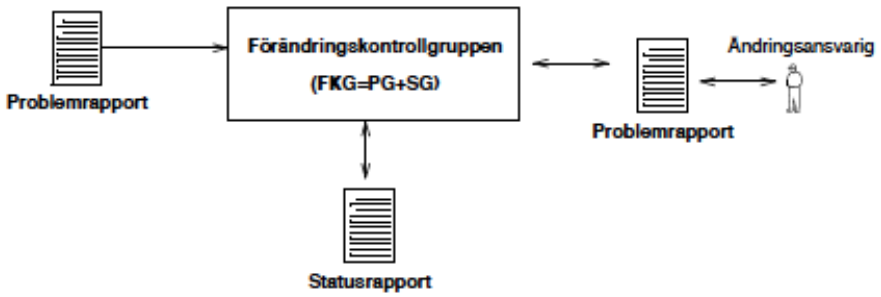
Ändringshantering ska ske från och med att ett dokument sätts i baseline. Innan ett dokument sätts i baseline behöver ändringshantering inte ske på något strukturerat sätt.

Till hjälp för ändringshanteringen finns två typer av rapporter, problemrapporter och statusrapporter.

Problemrapport Denna rapport upprättas när ett problem upptäcks och ges därefter till FKG. Problemrapporten kan genereras från utvecklingsarbetet eller från granskningar och test. Om FKG anser att problemet verkligen var ett problem (och inte t.ex. en enkel missuppfattning) gör FKG en utredning om vad som behöver ändras, ev. påverkan på andra konfigurationsenheter, vad det kan kosta att göra ändringen, etc. Utredningen resulterar i ett åtgärdsförslag.

Om resurser (tid och personal) finns tillgängliga, så beslutar FKG att ändringen ska utföras av en utsedd ändringsansvarig. Denne erhåller problemrapporten och gör ändringen i en kopia av konfigurationsenheten enligt åtgärdsförslaget, samt dokumenterar på problemrapporten exakt vad som ändrats och hur lång tid det tog. När ändringen är utförd skall FKG godkänna denna (eventuellt i samråd med kunden beroende på ändringens omfattning och typ, se nedan), och ge klartecken för versionsuppdatering och införande av ändringen i konfigurationsenheten. På så sätt blir problemrapporten en slags "stafettpinne" i ändringshanteringen.

Statusrapport För varje konfigurationsenhet ska FKG föra en statusrapport, d.v.s den ska ge en god bild över statusen för en specifik enhet. Det är



Figur 4.1: Problemrapporten skickas vidare under ändringshanteringen

lämpligt att upprätta en statusrapport för varje identifierad konfigurationsenhet. Statusrapporten är till för att FKG ska kunna hålla reda på vilka ändringar som är gjorda, samt utestående ändringar som ännu ej är införda. FKG behöver statusrapporten då problemrapporterna lämnas ut till ändringsansvariga och man vill när som helst kunna få en överblick över konfigurationsenheternas status. Statusrapporterna skapas samtidigt som första versionen av det aktuella dokumentet och skall börja användas när dokumentet sätts i baseline.

Figur 4.1 visar en översikt av ändringshanteringen.
Ändringshanteringen ska garantera att:

- en konfigurationsenhet endast blir ändrad enligt en överenskommelse om vilken form som förändringen ska ta.
- en ändring i en konfigurationsenheten endast införs då förändringen har blivit kontrollerad och godkänd.

En ändring tar tid att genomföra och går igenom ett antal “tillstånd”, se figur 4.2. Dessa tillstånd återspeglas i problemrapporten och statusrapporten.

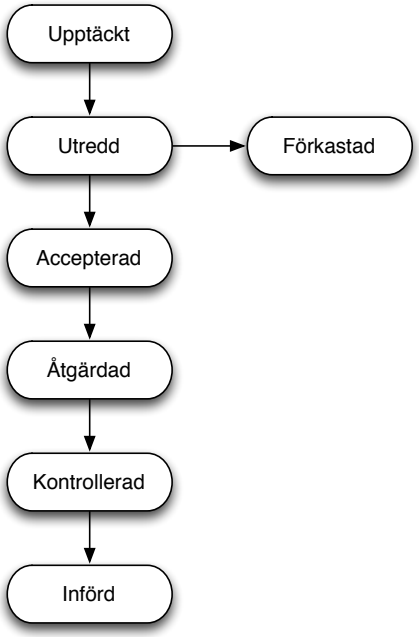
Förändringskontrollgruppen konsulterar kunden (via sektionschefen) om ändringen är av större karaktär. Detta gäller i följande fall:

- Alla ändringar av deadlines som påverkar kunden eller formella granskningar, dvs ändringar i SDP:n.
- Alla ändringar som har att göra med tolkningen av kundens önskemål, dvs ändringar i SRS:n.

När en konfigurationsenhet har blivit förändrad är det viktigt att hålla reda på vilka förändringar som har blivit gjorda. Förutom via statusrapporten, görs detta med en tabell, se figur 4.3, som anger vilka förändringar som har blivit gjorda i varje version av komponenten.

Om konfigurationsenheten är ett dokument, placeras förändringstabellen i början av ett dokument, och om det gäller en systemkomponent placeras tabellen i en kommentar till koden.

Problemrapporter och statusrapporter kan skötas i det elektroniska system som institutionen erbjuder i kursen (”ePUSS”). Det är även tillåtet att sköta



Figur 4.2: Tillstånden i livcykeln för ett problem/fel som kräver ändringsåtgärd

```
Dokumentnamn: STLDD
Ansvarig: SG
Datum upprättad: 120701
Nuvarande version: 1.1
Modifieringshistoria:
Vers.  Datum    Tid    Ansv.     Förändring       Anledning
0.1    120701   0955   SG        Skapades
0.2    120912   2315   UG1       Ny process Y     PDR.4
1.0    120913   1215   SG        baseline
1.1    120915   2115   UG2       Ny signal X      PR12
```

Figur 4.3: Exempel på förändringshistorik

detta i något annat system så länge man har en strukturerad process som man följer i hela projektet. Anreppssättet man väljer ska beskrivas i projektplanen.

4.5 Projektbibliotek

För att konfigurationshanteringen skall fungera, måste alla inblandade vara på det klara med hur dokument, rapporter och filer skall lagras. Det är viktigt att hela tiden hålla reda på var olika versioner finns. För detta ändamål bildas ett speciellt *projektbibliotek* som består av ett dokumentbibliotek och ett arbetsbibliotek.

4.5.1 Dokumentbibliotek

De dokument som produceras under projektet behöver behandlas på diverse möten och granskningar vilken innebär att de måste vara lätt åtkomliga för externa granskare och gruppens medlemmar. Dessa dokument lagras i *dokumentbiblioteket*, vilket kan ha följande innehåll:

- Dokument ingående i upprättad specifikationsbaseline (SBL).
- Dokument ingående i upprättad design- och testbaseline (DTBL).
- Dokument ingående i upprättad produktbaseline (PBL).
- Dokument rörande fel- och ändringshanteringen, både åtgärdade och icke åtgärdade.
- Granskningsprotokoll från de formella granskningarna.
- Mötesprotokoll och granskningsprotokoll från de informella granskningarna.

Projektgruppen ska under första veckan komma överens med sektionschefen om dokumentbibliotekets exakta innehåll och hur det ska publiceras. det kan t ex ske genom en hemsida eller en pärm med papperskopior. Det är viktigt att det är enkelt för granskaren att få tillgång till aktuellt material och tidigare baselines vid granskningar.

Dokument tillhörande dokumentbibliotek och som ej ingår i upprättad baseline får hanteras fritt av projektet. Projektledarna, som är ansvariga för dokumentbiblioteket, ska dock hela tiden veta var alla dokument finns och på förfrågan från sektionschefen eller granskaren ska dokumenten kunna uppvisas.

4.5.2 Arbetsbibliotek

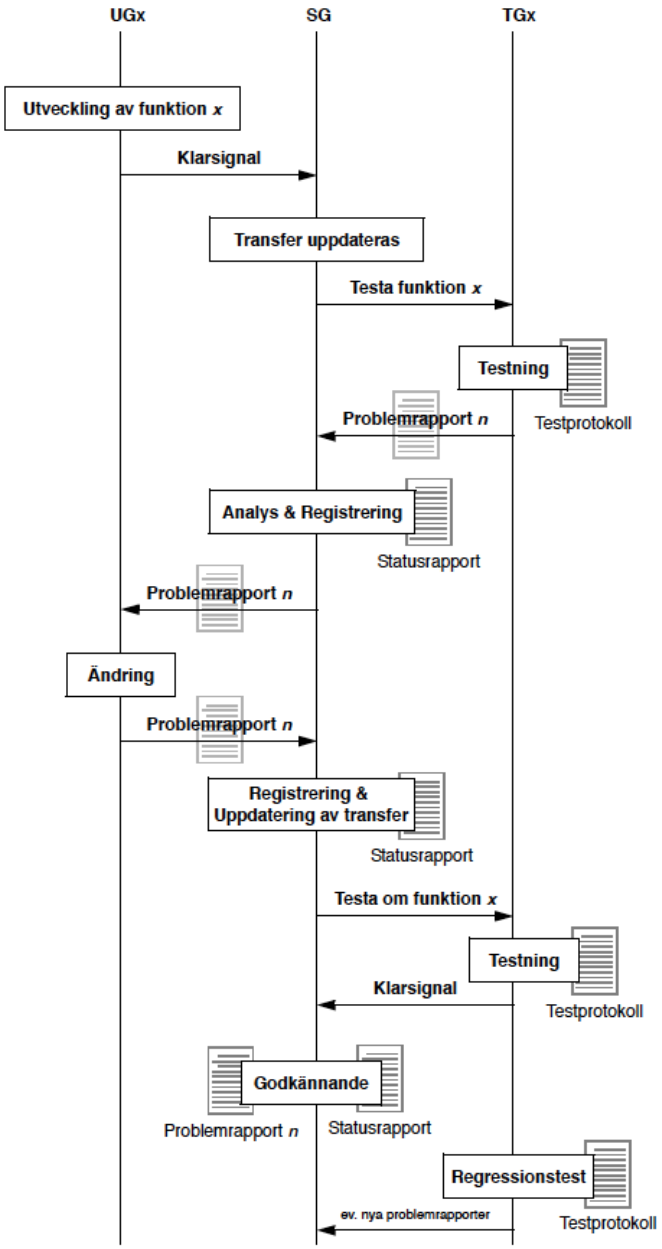
Alla dokument skall finnas tillgängliga i elektroniskt läsbar form under projektets gång. Dessa lagras i arbetsbiblioteket, som utgörs av filer som gruppens medlemmar kommer åt.

Projektet bestämmer själva hur detta ska gå till, t ex med hjälp av git/svn eller annat verktyg.

4.6 Arbetsgång i arbetet med lågnivådesignen

Under arbetet med implementationen, kan behov av ändringar uppstå. Hur denna ändringshantering skall gå till visas med ett exempel i figur 4.4. I detta exempel antas följande:

- Utvecklarna utvecklar en funktion x. Denna läggs sedan i "Transfer" som i detta exempel är den area i arbetsbiblioteket som innehåller den senaste testbara versionen.
- Vid testning av funktion x upptäcks ett fel,
- Analysen ger att felet måste åtgärdas,
- Ändringen medför att funktion x fungerar felfritt.



Figur 4.4: Figuren visar hur kommunikationen mellan de olika grupperna kan ske efter felupptäckt vid ett test.

Kapitel 5

Möten

Under projektet behöver olika typer av möten hållas, t.ex. för att koordinera arbetet, informera varandra, fatta beslut, lösa problem, etc. Projektgruppen behöver hålla interna möten (s.k. projektmöten), där delar eller hela projektgruppen träffas. Även möten med utomstående (sektionschef, experter och kund) behöver hållas för att rapportera och göra kvalitetsuppföljningar. Dessa olika möten beskrivs nedan.

5.1 Projektmöten

Möten ska hållas regelbundet, minst en gång i veckan. Projektledarna är ansvariga för att detta efterlevs. Projektledarna ska sammankalla till möten och agera ordförande och sekreterare under dessa, samtliga medlemmar i gruppen bör vara med på mötena.

Syftet med mötena är:

- Kontrollera att tidplanen följs.
- Kontrollera att riktlinjerna i utvecklingsplanen (SDP) följs.
- Diskutera problem för gemensamma och enskilda delar av systemet.
- Förbereda och följa upp formella granskningar.

Även informella möten är en form av projektmöte.

Mötena ska dokumenteras, i ett mötesprotokoll, och införas i projektbiblioteket. PG är ansvariga för detta.

Som komplement till projektmötena är det viktigt att ha tekniska möten, där delar av gruppen kan träffas och diskutera specifika tekniska problem och lösningar. Tekniska möten kan behövas för att t ex samordna kravspecifikationen (SRS) med testspecifikationen (SVVS) (troliga deltagare systemgrupp och testgrupp) eller tekniska diskussioner rörande hur man ska implementera systemet (troliga deltagare systemgrupp och någon eller några utvecklingsgrupper).

5.2 Möte med sektionschef

Möte ska hållas minst en gång per vecka. Deltagare är, förutom sektionschefen, projektledargruppen. Mötenas syfte är att:

- Ge sektionschefen en helhetsbild över läget i projektet.
- Följa upp att tidplanen och riktlinjerna i utvecklingsplanen (SDP) följs.
- Ge möjlighet för projektledarna att diskutera icke-tekniska problem med sektionsschefen och andra projektledare.

Projektledarna förbereder inför varje möte en kort rapport med information om vad som går bra i projektet och vad som är eventuella problem i projektet.

Mötena behöver ej dokumenteras formellt. Det är dock viktigt att projektledargruppen informerar projektgruppen om information som erhålls under mötena. Informationen kan t ex gälla speciella önskemål vad avser utformningen av något dokument.

5.3 Expertmöte

Följande experter finns tillgängliga för projektet:

- Kravexpert – SRS-frågor
- Testexpert – SVVS-, SVVI- och SVVR-frågor
- Designexpert – STLDD- och SDDD-frågor

Expertmöten hålls vid behov och efter överenskommelse med respektive expert. Dock ska ett obligatoriskt möte med respektive expert hållas inför de formella granskningarna av respektive dokument.

De obligatoriska mötena sker på tid och plats enligt kursprogrammet eller enligt överenskommelse. Expertmötena behöver ej dokumenteras formellt.

Kapitel 6

Granskningar

Innan ett dokument godkänns, är det klokt att gå igenom detta noga. Vid en granskning (eng. review, inspection) görs en strukturerad genomläsning av ett dokument och ett granskningsmöte hålls enligt vissa regler där beslut om åtgärd eller godkännande fattas, baserat på de kommentarer som hittats vid genomläsningen.

Granskningarna är till för att säkerställa att den färdiga produkten blir den som kunden har beställt. Granskningarna används alltså som en metod för validering och verifiering. Granskningarna ska se till att eventuella fel och skillnader gentemot kundens krav upptäcks på ett så tidigt stadium av utvecklingen som möjligt. Ju senare dessa upptäcks desto dyrare blir de att rätta. Vidare utgör granskningarna en hållpunkt som visar hur utvecklingen av produkten ligger till tidsmässigt gentemot den utvecklingsplan som bestämts.

I projektet finns två typer av granskningar: *formella granskningar*, där en representant från kunden medverkar, samt *informella granskningar*, där projektet granskar ett dokument internt utan kundens medverkan.

De informella granskningarna är viktiga dels inför formella granskningar och dels inför det fortsatta arbetet. Dokument som har granskats lägger basen för fortsättningen i projektet och därmed är granskningarna oerhört viktiga för att säkerställa projektets framåtskridande.

Granskningarna ska dokumenteras, i ett granskningsprotokoll, och detta ska införas i projektbiblioteket. PG är ansvariga för att granskningar planeras och genomförs. Författare och granskare har efter informell granskning gemensamt ansvar för att dokumenten dels är klara för formell granskning och dels att dokumenten är lämpliga att bygga fortsatt arbete på.

Om en granskning upptäcker ett fel i ett dokument som ingår i baseline, upprättas problemrapporter och ändringen följs upp i statusrapporten för dokumentet.

6.1 Granskningsmetod

Granskning kan ske på olika sätt. I detta projekt är alla granskningar uppdelade i följande faser:

Planering Här planeras granskningen t ex med avseende på vad som ska granskas och vem som ska granska vilka delar av dokumenten. det är även nöd-

vändigt att boka tid för granskningen och att skicka ut information till alla granskare om granskningen.

Individuell förberedelse Här genomför alla granskare en individuell granskning med målet att hitta fel och oklarheter i det granskade dokumentet. Till stöd för detta kan man t ex ha en checklista vid granskningen.

Granskningsmöte Vi granskningsmötet gås alla fel igenom som upptäckts vid den individuella granskningen. Moderatorn är ansvarig för att gruppen hinner igenom alla fel inom mötestiden, vilket innebär att han/hon måste bryta diskussioner då dessa drar ut på tiden. det är viktigt att mötet fokuserar på att identifiera fel och att eventuella diskussioner t ex om lika lösningsförslag skjuts till efter mötet.

I början av granskningsmötet ska moderatorn klargöra följande frågor:

- Är mötet granskningsmässigt? (Är rätt personer närvarande och har de läst på?)
- Är dokumenten granskningsmässiga? (Är det överhuvudtaget någon mening med att gå igenom dokumenten?)

I slutet av granskningsmötet fattas ett beslut om fortsättningen:

- Dokumenten godkännes som de är.
- Dokumenten godkännes efter ändringar, där anmärkningarna åtgärdas. Moderatorn kontrollerar att dessa utförs och godkänner.
- Anmärkningarna ska åtgärdas, varefter *omgranskning* planeras och utförs vid överenskommet tillfälle.
- Dokumenten underkänns helt och ska skrivas om, följt av ny granskning enligt överenskommelse.

Uppdatering Dokumenten uppdateras enligt beslut på granskningsmötet.

Följande roller är alltså inblandade i granskningen:

Koordinator Denna roll är ansvarig för planeringen av granskningen.

Moderator Denna roll leder granskningsmötet (se ovan).

Sekreterare Under mötet ska sekreteraren notera de anmärkningar mötet enas om enligt mallen för granskningsprotokoll. Vid minsta tvekan ska sekreteraren läsa upp anmärkningar för de närvarande och fråga om de uppfattats korrekt. Om sekreteraren inte hinner med skall han/hon be moderatorn att vänta.

Granskare är ansvarig för att individuellt granska dokumentet enligt instruktioner och för att delta på granskningsmötet och kunna förklara de fel han/hon rapporterat.

Författare är den eller de personer som har producerat det granskade dokumentet. Dessa personer ska delta på granskningsmötet för att kunna förklara vid eventuella oklarheter.

En person kan ha mer än en roll vid en granskning. T ex kan koordinatorn vara samma person som sekreteraren.

6.2 Formella granskningar

Vid formella granskningar gäller följande:

- PG är koordinator för granskningen. Detta innebär t ex att PG är ansvariga för att boka tid med granskaren.
- Extern granskare, som är utsedda av institutionen, är ensam granskare av alla dokument som ska granskas.
- En representant från institutionen agerar moderator.
- Koordinatorn levererar alla dokument till granskaren senast enligt deadline i kursprogram.
- Granskaren levererar grankningskommentarer till hela gruppen innan granskningen.
- En av projektledarna är sekreterare. Då det bara finns en granskare räcker det att sekreteraren noterar avvikelser från granskarens protokoll som man kommer överens om på mötet.

Granskningsmötet omfattar normalt ca 60 minuter. Samtliga gruppmedlemmar bör delta. Det är då viktigt att samtliga författare läser igenom kommentarerna och tänker igenom dem som förberedelse innan mötet. Vid granskningsmötet behövs då enbart kommentarer tas upp som någon projektdeltagare har frågor om eller som granskaren vill förtydliga. Om granskningsobjektet godkänns eller ej avgörs i slutet av mötet av moderatorn.

Det finns tre formella granskningar i projektet: SSR, PDR och PR. PR sköts dock lite speciellt eftersom mötet infaller efter kursen.

6.2.1 Software Specification Review (SSR)

Här granskas SDP, SRS och SVVS. Dessutom granskas dokumentbibliotekets innehåll och struktur, samt övrig rapportering.

Granskningen sker under vecka 3 och godkända dokument införs i Specifikationsbaseline (SBL).

6.2.2 Preliminary Design Review (PDR)

Här granskas SVVI och STLDD. Dessutom granskas dokumentbibliotekets innehåll och struktur, samt övrig rapportering.

När granskningen godkänt STLDD och SVVI införs de i Design and Test Baseline (DTBL).

6.2.3 Product Review (PR)

När produkten anses klar levereras dokumentbiblioteket till representant för kunden. Dokument godkända av projektet införs i Product Baseline (PBL). Det senare innebär att projektet skall sätta samtliga dokument ifrån fas 4 (SVVR, SSD och PFR) i baseline innan dokumenten lämnas till kunden, då projektet inte förväntas uppdatera dokumenten efter leverans till kunden.

Acceptans av produkten (dokument och program) görs genom granskning av alla tillhörande dokument, samt utförande av ett acceptanstest. Resultatet av Product Review levereras till gruppen, samt gås igenom på ett frivilligt acceptansmöte.

6.3 Informella granskningar

Vid informella granskningar finns kunden ej representerad. Inför varje formell granskning ska projektgruppen internt genomföra en informell granskning vars syfte är att gå igenom varje delgrupps material som om det vore en formell granskning. Tidpunkt för granskningen ska väljas så att inlämningen till formell granskning ej blir försenad, denna bestäms av projektledargruppen i samråd med projektdeltagarna och en koordinator utses.

Det ska också ske en informell granskning efter fas 3, där ingen formell granskning finns. Under denna ska SDDD gås igenom, så att eventuellt kvarvarande fel kan lösas.

Följande är viktigt att notera med de informella granskningarna:

- Koordinatorn är ansvarig för att planera granskningarna så att det finns lämpliga granskare att tillgå och att alla dokument blir ordentligt granskade. Planeringen ska innefatta en lista över planerade granskningar och en lista över deltagare i dessa granskningar.
- Rollerna i samband med granskningarna är viktiga även vid den informella granskningen.
- Alla projektdeltagare förväntas inte deltaga i samtliga informella granskningar, utan här är det viktigt att man följer koordinatorns plan, se ovan.
- Det är viktigt att betona att avsikten inte är att klaga på varandra, utan att hjälpa varandra till ett lyckat projekt. Vidare är det väsentligt att författarna inte intar en försvarsställning, utan ser granskningen som en möjlighet att lyfta kvaliteten på sitt arbete ytterligare.

6.4 Granskningar under utvecklingen

De granskningar som beskrivs ovan görs för att godkänna hela dokument eller koden som går vidare till testfasen. Ofta gör man granskningar efter hand innan man lägger in kod i ”main branch”, t ex med verktyg som Gerrit. Detta är såklart inget som hindrar att man även granskar dokumenten enligt ovan.

Kapitel 7

Test

När en produkt är färdig behöver den kontrolleras, så att man är säker på att den uppfyller kundens ursprungliga behov och önskemål. Detta kallas validering (eng. validation). För att säkerställa att en produkt uppfyller dess specifikation utförs en verifiering. (eng. verification). Vid verifiering kontrolleras att produkten är korrekt utvecklad. Man kan säga att

- validering besvarar frågan "Är det rätt system?" och
- verifiering besvarar frågan "Är systemet rätt?".

Validering av systemet sker ofta av kunden genom granskning och s.k. *acceptanstest*. Verifiering av systemkomponenter görs ofta genom testning, medan verifiering av dokument sker genom granskning.

Testning innebär att ett exekverbart testobjekt (ett system eller en del av ett system) undersöks för att se om det uppfyller specifikationen för objektet (verifiering) eller kundens önskemål (validering). Testning kan delas upp i två huvudgrupper:

White-box testing där man utgår från kodens interna struktur när man gör testfall och försöker testa så att man t.ex. går igenom alla rader minst en gång, eller alla vägval i programflödet.

Black-box testing där man har ett externt perspektiv d.v.s testningen är baserad på indata och utdata. Man väljer testfall bland de kombinationer av indata som kan förekomma. Resultatet av exekveringen jämförs mot en specifikation.

Syftet med testen är alltså att kontrollera att samtliga krav som specificerats i kravspecifikationen är uppfyllda och att produkten följer designen.

7.1 Testfaser

I projektet sker ett antal typer av test:

Enhetstest utförs av utvecklingsgruppen efter hand som kod tas fram. Här utnyttjas både "white-box"- och "black-box"-testning

Funktionstest där enskilda funktioner testas enligt en funktionstestspecifikation och tillhörande funktionstestinstruktioner.

Systemtest där ett komplett system testas för att se hur flera funktioner och tjänster samverkar enligt en systemtestspecifikation. Testen genomförs i enlighet med de framtagna systemtestinstruktionerna.

Regressionstest När nya funktioner har införts eller gamla har ändrats (t.ex. på grund av en felrättning) behöver "gamla" funktioner testas, för att säkerställa att ändringen inte påverkat något som tidigare godkänts i test.

Acceptanstest under vilket kunden utför ett urval av ovanstående tester samt ett par för leverantören okända testfall, med syfte att validera systemet.

Enhetstest kan även genomföras som kodgrankning inom projektgruppen. Funktionstest, systemtest och regressionstest sker enligt "black-box"-testning. Funktionstest, systemtest och regressionstest ska utformas, dokumenteras och utföras av testgrupperna, medan acceptanstest utförs av kund.

7.2 Regressionstest

Regressionstest innebär att man testar om "gamla" funktioner efter att ändringar eller tillägg gjorts.

Under regressionstestet ska bland annat de test som finns framtagna för ett eventuellt grundsystem åter testas (dock bara om de är tillämpliga med de nya funktionerna).

Ändringar under utvecklingens gång kräver också att regressionstest görs, så att ändringar inte infört fel i delar som redan är klara. För att inte behöva köra igenom alla tidigare test när man gjort en ändring, så specificerar man i SVVS ett antal regressionstest för varje funktion. Dessa test ska köras varje gång en ändring eller ett tillägg gjorts. Dock ska mellan att sista ändring gjorts i systemet och själva projektinlämningen, samtliga testfall köras igenom utan att fel hittas.

Regressionstestet beskrivs, utförs och dokumenteras av testgruppen.

Ej godkänt test dokumenteras i en problemrapport och överlämnas till förändringskontrollgruppen, FKG.

Om det finns möjlighet att automatisera regressionstesten så kan i många fall mycket tid sparas.

7.3 Acceptanstest

Under acceptanstestet väljer kunden ut ett antal testfall ur ovanstående tester samt ett antal, för leverantören, okända testfall. Dessa tester baseras på respektive projekts SRS.

Kapitel 8

Dokumentinstruktioner

Under projektets gång kommer ett antal dokument att skapas och användas. Figur 8.1 visar hur dessa dokument beror av varandra.

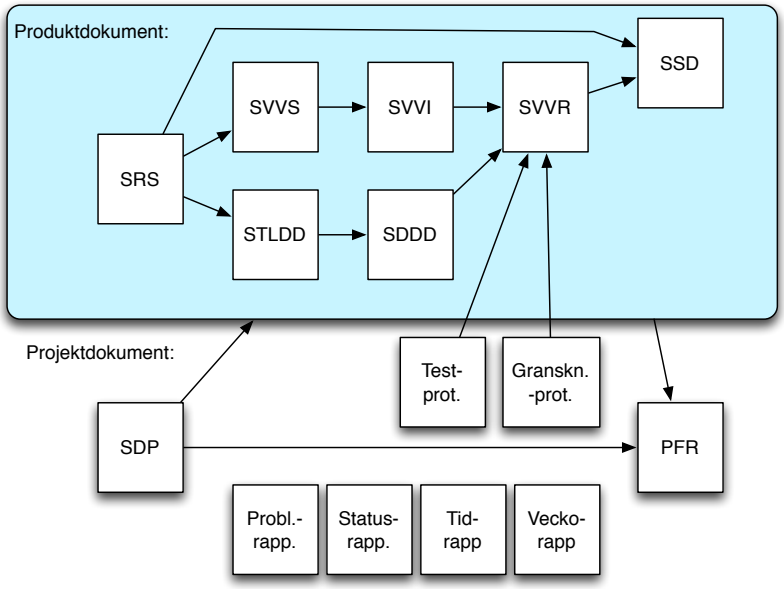
Det finns två huvudtyper av dokument, produktdokument och projektdokument, se figur 8.1. Produktdokument beskriver olika aspekter av produkten och består som dokumentation även efter det att projektet är avslutat. Projektdokument är primärt till för det pågående projektet, och används endast som erfarenhetsbas efter det att projektet är avslutat.

En viktig aspekt vid dokumentation är att man refererar till information som finns i andra dokument på rätt sätt. Generellt gäller att det är tillåtet att i produktdokument referera till produktdokument men inte till projektdokument från andra projekt. Motivet bakom detta är att produktdokument har en längre livstid och tillhör organisationen. Projektdokument har i första hand en livstid som är identisk projektets löptid.

8.1 Allmänna dokumentkrav

Följande allmänna krav gäller för produktdokumenten SRS, SVVS, SVVI, STLDD, SDDD, SVVR, SSD, samt projektdokumenten SDP och PFR:

1. Dokumenten ska följa beskrivningarna i kapitel 2 om inget annat anges i detta kapitel.
2. Dokumenten ska finnas tillgängligt i dokumentbiblioteket efter informell granskning.
3. Bilder/figurer ritas med valfritt datorbaserat ritverktyg.
4. Valfri ordbehandlare får användas.
5. Alla dokument ska vara på samma språk (svenska eller engelska).
6. Framsidor ska vara enhetliga och innehållande:
    - Namn på dokumentet
    - Gruppnummer
    - Ett dokumenthuvud innehållande:



Figur 8.1: Dokument och relationer mellan dokument

- Ansvarig för dokumentet
- Uppgjord (författare)
- Datum
- Dokumentnummer
- Dokumentversion

8.2 Software Development Plan (SDP)

SDP ska innehålla en beskrivning av utvecklingsmodellen (referenser till utvecklingsmodellen kapitel 2 och beskrivning av eventuella anpassningar).

8.2.1 Projektorganisation

SDP ska även innehålla en beskrivning av projektorganisationen, där det ska vara tydligt vilka ansvarsområden som finns och vem som har ansvar för vad. Beskriv t ex vem som är projektledare, vem som är med i systemgruppen, vem som är med i testgruppen, vem som är med i de olika utvecklingsgrupperna etc. Beskriv även vilka externa intressenter det finns till projektet.

8.2.2 Tidplan

I tidplanen ska det finnas en detaljerad nedbrytning av det arbete som ska ske i aktiviteter. Det ska också finnas skattningar av arbetstid (dvs “effort”), ledtid och datum för när aktiviteter ska vara färdiga. Det betyder att man ska kunna utläsa minst följande:

1. Skattad tidsåtgång för varje fas (hur fördelas arbetsinsatsen över projektets faser?).
2. Skattad start- och slutdatum för varje fas (när blir olika delar klara?).
3. Skattad tidsåtgång för varje dokument (vad kostar varje del?).
4. Skattad start- och slutdatum för varje dokument (när blir olika dokument klara?).
5. Skattad tidsåtgång för olika aktiviteter och aktivitetstyper i respektive fas (vad lägger man tiden på?). (Möten, granskningar, ändringshantering, rapportering, etc.)
6. Skattad tidsåtgång för varje grupp, uppdelat per vecka (går det in på en 40-timmarsvecka?).
7. En kalenderplan där man kan se vad varje grupp ska göra varje vecka (vem ska göra vad och när?). Detta kan t ex åskådliggöras i ett "Gantt-schema".
8. Det ska även vara tydligt vilka möten som är planerade under projektet, både interna möten och möten där externa intressenter är med.

Ange även i projektplanen vilka metoder ni använt för att göra skattningar av tid och kostnad, samt vilka de största osäkerheterna är med skattningarna.

8.2.3 Uppföljning och kvalitetsutvärdering

Det ska finnas en del i projektplanen som beskriver hur uppföljning, t ex av tidplanen, sker under projektet, samt vad som händer om arbetet inte verkar gå enligt plan. Det ska också finnas en beskrivning av de rutiner som finns för kvalitetsutvärdering under projektet.

8.2.4 Riskanalys

I projektplanen ska även resultatet av en riskanalys för projektet presenteras. Ange hur riskanalys utförts i projektet, samt de viktigaste riskerna som identifierats. rapportera åtminstone följande för varje rapporteras risk: skattad sannolikhet (t ex, låg, medel, hög), skattad effekt (t ex, låg, medel, hög), möjliga indikatorer på att risken förvekligas, samt exempel på lösningar om risken förverkligas.

8.3 Software Requirements Specification (SRS)

Detta dokument definierar de krav som kunden och leverantören har kommit överens om att systemet ska uppfylla. SRS utgör således en mycket viktig del i kontraktet mellan kund och leverantör, och kvalitetskraven på SRS är därför höga. Följande kvalitetsattribut hos SRS är extra viktiga och ska kontrolleras på både informella och formella granskningar:

Korrekthet – överensstämmer kraven med kundens önskemål?

Fullständighet – är alla kundens krav med?

Verifierbarhet – är kraven uttryckta på ett testbart sätt?

Entydighet – finns det bara en tolkning av kraven?

Motsägelsefrihet – är kraven fria från konflikter?

Spårbarhet – är det lätt att referera till varje krav?

Organiserad – är det lätt att hitta bland kraven?

Dessa frågor kan användas som checklista under kravgranskningar.

För att åstadkomma spårbarhet ska alla utsagor som ställer krav på systemet vara numrerade med ett unikt löpnummer, å att man lätt kan referera till ett specifikt krav. Löpnumren struktureras lämpligen in i SRS:ens kapitelstruktur så att kraven i t ex kapitel 4.1 börjar med krav nummer 4.1.1, 4.1.2 osv. I kapitel 4.2 finns kraven 4.2.1, 4.2.2 osv. Om ett krav utgår i en senare version av SRS:en (*efter baseline*) behöver man inte numrera om alla kraven, utan skriva "Krav 4.2.1 utgått".

För att SRS ska vara lätt att hitta i, ska kraven vara uppdelade i funktionella och ickefunktionella krav. Ett funktionellt krav behandlar systemets funktion relaterat till t ex indata och utdata. Ett icke-funktionellt krav behandlar begränsningar på systemet, t ex prestanda, tillförlitlighet, minnesbegränsningar, effektivitet och användarvänlighet.

De funktionella kraven sorteras lämpligen under rubriker för olika funktionalitet hos det utvecklade systemet.

För att underlätta entydigheten måste alla termer som används, vara definierade i ett speciellt terminologi-kapitel.

För att underlätta fullständigheten ska relevanta felsituationer och onormala beteenden ingå i kraven. Det är också viktigt att definiera vad som händer i alla kombinationer av olika funktionalitet. T ex, vad händer i ett telefonsystem om man med funktionaliteten underhåll tar bort en abonnent som man med funktionaliteten vidarekoppling vidarekopplat en telefon till?

Då systemet bygger på en datamodell är det lämpligt att även denna finns med i kravspecifikationen.

För att underlätta verifierbarheten är det viktigt att testgrupperna är med i kravarbetet och ser till att det som skrivs i SRS går att testa. Icke verifierbara krav är oftast meningslösa. Synkroniseringen med SVVS:en är alltså mycket viktig.

Hela utvecklingen är beroende av att SRS har en hög kvalitet. Om man upptäcker felaktigheter i kraven sent i utvecklingen kan dessa bli mycket dyra att åtgärda. Eftersom kravspecifikationen är ett så viktigt dokument i projektet och den förhållandevis begränsade kommunikation som finns med kunden efter fas 1 i projektet, är det lämpligt att inkludera förhållandevis mycket detaljer i specifikationen. Exempel på delar som bör vara med är datamodell, beskrivning av användargränssnitt i termer av vilken information som matas in och visas, vad som händer vid felaktig inmatning, etc.

För att utvecklarna lättare ska förstå vad som menas, är det bra att inkludera motiveringar till kraven. Det är viktigt att förstå varför ett krav ser ut som det gör, när man i designen skall göra kostnadseffektiva avvägningar.

8.4 Software Verification and Validation Specification (SVVS)

Tanken med SVVS:n är att beskriva hur verifiering och validering ska gå till i projektet. Detta innebär att både granskningar och traditionell test ska beskrivas. SVVS:en beskriver alltså planen för hur verifiering och validering ska gå till, t ex i termer av *vilka* testfall som ska köras, men detaljer om hur de olika testfallen ska utföras finns i stället i SVVI:n.

När det gäller granskningar ska det i SVVS:n t ex beskrivas vilka typer av granskningar som ska utföras i projektet, vilka dokument som ska granskas, samt när granskningar ska ske.

När det gäller test så ska SVVS:n beskriva:

- Vilka typer av test som ska utföras, t ex enhetstest, funktionstest, etc.
- När olika typer av test ska ske.
- Vilka olika målmiljöer som finns för test i projektet, t ex simulerad miljö och verklig miljö.
- När test i olika miljöer ska ske, samt vilka typer av test som ska ske i olika miljöer.
- Kopplingen till krav, dvs det måste finnas en spårbarhet från testfall i SVVS:en till krav i SRS:en. T ex kan man efter varje testfall ange vilka krav som testas. (Alla krav ska testas av minst ett testfall. Alla testfall ska testa minst ett krav.)

Följande delar ska finnas med som appendix till SVVS:

Funktionstestspecifikation innehåller en numrerad lista med en kort beskrivning av alla funktionstest.

Systemtestspecifikation innehåller en numrerad lista med en kort beskrivning av alla systemtest. Systemtesten innefattar "interaktionstest", dvs test där flera funktioner interagerar.

Regressionstestspecifikation beskriver vilka test som man avser köra som regressionstest för eventuell grundfunktionalitet, samt för respektive av de nya funktionerna.

8.5 Software Verification and Validation Instructions (SVVI)

Testinstruktionsdokument inleds med en allmän introduktion samt en beskrivning av testutförandet. Testutförandet beskriver vilka test som ska utföras med hjälp av granskning respektive vilka som ska göras med exekverande testning. Detta ska vara uppdelad i system- funktion- och regressionstest. Därefter ska SVVI innehålla två appendix: funktionstestinstruktioner och systemtestinstruktioner.

Testinstruktionerna ska innehålla en kort inledning som beskriver testfallet och det förväntade resultatet. Vidare ska inledningen definiera vilka användare

som är inblandade. De detaljerade testinstruktionerna ska sedan följa under tre rubriker:

- Uppsättning av systemstatus t.ex. tomt system, administratören inloggad
- Testfall t.ex. administratören lägger till en ny användare A
- Uppföljning av testfall t.ex. A inloggad

Varje testinstruktion ska bara innehålla ett testfall, dock kan likvärdiga testfall finns på samma sida (för att spara papper). Testfallen får inte innehålla referenser till andra testfall.

Formaten för testinstruktionerna skall överensstämma med grundsystemets testinstruktioner. Numrering av testfallen är väsentlig så att man enkelt kan spåra en testinstruktion till motsvarande testspecifikation.

SVVI kan även innehålla en lista på initieringsfiler för att få systemet i ett visst läge, tex ett tomt system med en administratör men inga användare eller projektgrupper.

8.6 Software Top Level Design Document (STLDD)

STLDD ska innehålla en högnivådesign av systemet. Det finns inga absoluta regler för hur designen ska beskrivas men det är viktigt att systemets *arkitektur* beskrivs så att det är tydligt vilka delsystem som systemet är uppdelat i och hur gränssnitten mellan delsystemen ser ut. Arkitekturbeskrivningen ska kunna användas för att alla intressenter ska förstå hur systemet är uppbyggt och för att kunna göra en första analys av systemet med avseende på kvalitetskrav.

Det ska dessutom finnas en mer detaljerad design som beskriver systemets komponenter. Beroende på vilken typ av system som utvecklas gäller följande:

System utvecklat i SDL Varje process ska ha ett eget kapitel innehållande beskrivningar av alla signaler till och från processen och vad processen har för uppgift.

För att tjänsterna ska bli så oberoende av varandra som möjligt skall varje funktion bestå av en egen process som, tar emot och kontrollerar siffersekvensen, utför den beställda funktionen, t.ex skickar en begäran om ändring i databasen, samt kontrollerar om det gick bra t.ex genom att kontrollera svaret från databasen.

STLDD ska innehålla minst 3 MSC:er för varje funktion inom respektive tjänst, ett för "normalfallet" samt minst två situationer där olika "felfall" uppstår. Ju fler MSC:er som görs desto lättare är det att implementera tjänstens funktionalitet. Tänk också på att "ju mer lika" lösningsförslagen för de olika tjänsterna blir, desto lättare att implementera och hitta eventuella fel blir det. Variabel- och signalnamn ska vara på engelska och självförklarande. Tänk också på att utnyttja gamla signaler i den mån det går, t.ex. ServiceOK kan användas som OK-signal för samtliga tjänster.

Objektorienterad design, implementerad t ex i Java I den mån det är lämpligt kan UML användas för att beskriva designen. Varje klass och varje publik metod ska beskrivas, vilket, t ex kan göras med hjälp av "javadoc". Dessutom ska ett klassdiagram produceras.

För varje viktig funktionalitet hos systemet ska minst två sekvensdiagram beskrivas, lämpligen ett för normal användning av systemet och ett för en annan användning av systemet t.ex. när man gör “fel".

Om en databas används, så ska strukturen på databasen beskrivas i detalj. Detta kan göras i ett ER-diagram som också visar alla “fields”. Det kan också vara lämpligt att skriva ut SQL-koden som genererar alla tabeller i databasen.

8.7 Software Detailed Design Document (SDDD)

SDDD är den mest detaljerade nivån av designen. Om utveckling sker i Java så består SDDD:n av själva java-koden och om utveckling sker i SDL så består SDDD:n av SDL-graferna (lågnivådesignen inkl. SDL-graferna ur STLDD).

Generellt gäller följande för koden:

- Alla variabelnamn ska vara på engelska och självförklarande.
- Tydliga kommentarer på engelska skall ges vid behov.

För java-kod gäller att den ska följa följande standard:
`http://www.geosoft.no/development/javastyle.html`
som finns under följande sida: `http://www.geosoft.no`

8.8 Software Verification and Validation Report (SVVR)

SVVR ska innehålla resultat av och kommentarer om de moment som beskrivits i SVVS och SVVI. Följande delar ingår som appendix till SVVR:

- Funktionstestresultat.
- Systemtestresultat.
- Regressionstestresultat.
- Granskningsprotokoll från alla granskningar.

8.9 System Specification Document (SSD)

I SSD ska systemets olika dokument “knytas samman” till ett system. SSD ska innehålla en beskrivning av det levererade systemets olika delar, versioner etc.

SSD ska även innehålla en beskrivning av eventuella skillnader gentemot SRS. Alla större förändringar av SRS eller förändringar av SRS som har betydelse för kunden skall beskrivas. Detta avser förändringar som är gjorda sedan specifikationsbaseline (SBL - Specification Baseline). Om några krav inte kunnat uppfyllas skall detta tydligt anges med hänvisning till relevanta testfall och testprotokoll.

Det är mycket viktigt att det i SSD står var man kan hitta alla dokument och filer, samt hur man startar systemet.

8.10 Project Final Report (PFR)

I samband med projektet ska data samlas in för att bl.a. följa upp nedlagd tid och antal fel. Denna information är en bra grund att ha då man ska starta ett nytt projekt, och dokumenteras därför i en slutrapport. Genom att analysera denna rapport kan man då göra en bra planering både med avseende på tid i allmänhet och resurser som behövs för att rätta de fel som man trots allt gör. I slutrapporten är det således viktigt att man sammanställer insamlad data på ett lämpligt sätt samt att man drar slutsatser utifrån dessa.

Slutrapporten bör ha följande avdelningar:

Historisk överblick över projektet Vad är det som har hänt? Siffror, tabeller och diagram. Jämförelser och uppvisande av data.

Utvärdering av vad som gick bra/dåligt Varför ser det ut som det gör? Analys av de siffror man samlat in. Varför tog det så lång tid i den fasen? Varför blev den modulen så snabbt klar?

Förbättringsförslag Hur ska man göra för att förbättra de positiva trenderna och få bort de negativa trenderna? Processförbättring mm.

För alla data som skattades i *tidplanen* (se kap 8.2.2) ska verkligt utfall redovisas. Om tidplanen uppdaterats under projektets gång så ska jämförelse ske med de ursprungliga värdena.

Eftersom dokumentet skrivs innan sista veckan är färdig så finns inte metricsdata tillgänglig för sista veckan i projektet. Detta kan lösas antingen genom att inte ta med sista veckan i analysen, eller genom att använda skattade värden för sista veckan.

Slutrapporten ska omfatta *max 20 sidor* (exklusive försättsblad, innehållsförteckning, etc.). Målgruppen för en slutrapport har ofta ont om tid, samtidigt som det är viktigt att rapporten blir läst. Alltså ska omfånget vara begränsat och gäller att välja vad man presenterar och sättet man gör det på.

8.11 Individuell rapport

Den individuella rapporten är ett kursdokument i kursen på LTH och inte ett projekt- eller produktdokument, vilket betyder att den inte presenteras i detta kompendium. Instruktioner finns på kursens hemsida.

8.12 Övriga rapporter och protokoll

För att underlätta datainsamlingen och konfigurationshanteringen har vi skapat ett antal blanketter som skall användas av projekten (se bilaga A). Dessa är:

- Tidrapport
- Veckorapport*
- Granskningsprotokoll
- Testprotokoll

Tabell 8.1: Aktiviteter

Nummer	Aktivitet	Specificering
11	SDP	Arbete med dokument med bilagor.
12	SRS	
13	SVVS	
14	STLDD	
15	SVVI	
16	SDDD	
17	SVVR	
18	SSD	
19	Slutrapport	
21	Funktionstest	Arbete med testning
22	Systemtest	
23	Regressionstest	
30	Möte	Gruppmöte, expertmöte, etc
41	Föreläsning	Inlärning, "kurstid"
42	Övning	
43	Terminalövning	
44	Självstudier	
100	Övrigt	

- Problemrapport*
- Statusrapport*

Blanketterna är konstruerade så att det skall vara lätt att spåra relaterad information. För att kunna använda dessa på ett riktigt sätt ges i följande delkapitel en beskrivning av innehållet på respektive blankett. Exempel på användning ges i kapitel 9.20 och 4. De blanketter som i punktlistan ovan är markerade med asterisk (*) finns även i elektronisk version i verktyget ePUSS, som kan nås via kursens hemsida.

8.12.1 Aktivitetsspecifikation

I tidrapporter och veckorapporter skall tiden man arbetar med olika aktiviteter redovisas. För detta ändamål har ett antal nummer och typer definierats som beskrivs nedan och sammanfattas i tabell 8.1.

8.12.2 Aktivitetstyper

Det finns fyra olika aktivitetstyper: U, I, F och O:

U (Utveckling och dokumentation): All den tid som läggs på att skapa saker från grunden tillhör denna aktivitetstyp. Det kan vara testfallsutveckling, design, implementering eller projektplanering. Utvecklingen och dokumentationen omfattar den tid som används till dessa aktiviteter fram tills dess att man anser sig färdig för att granska eller testa det man har producerat. Om det efter granskning uppkommer något fel eller

om något har blivit glömt så tillhör den tiden inte denna aktivitetstyp utan omarbetning, förbättring och felreparation.

I (Informell granskning): Förberedelsetid och mötestid för informell granskning. Den tid som ägnas åt att förbereda granskningen genom att läsa igenom material och liknande skall tas upp här. Likaså skall mötestiden inkluderas.

F (Formell granskning): Förberedelsetid och mötestid för formell granskning. Den tid som ägnas åt att förbereda granskningen och liknande skall tas upp här. Likaså skall mötestiden inkluderas.

O (Omarbete): Den tid som går åt att göra förändringar skall inkluderas här. Det kan vara saker så som felrättning, regressionstest p.g.a. felrättning eller omtestning eller om det t.ex. fanns syftningsfel i ett dokument.

8.12.3 Felklasser och allvarlighet

I granskningsprotokoll och problemrapporten ska felen man finner redovisas och klassificeras. För detta ändamål har ett antal felklasser och allvarlighetsgrader definierats.

De felklasser som är definierade är listade i tabell 8.2.

Det finns tre grader av allvarlighet av fel, A, B och C:

A (Allvarligt fel): Felet omöjliggör fortsatt exekvering. Dessa fel ska alltid korrigeras.

B (Mindre fel): Felet hindrar bara den testade funktionen. Exekveringen kan fortsätta. Dessa fel ska alltid korrigeras.

C (Obetydligt fel): Funktionen kunde utföras men med smärre problem.

Tabell 8.2: Felklasser

	Nr	Feltyp	Eventuell beskrivning
Generella	11	Stavfel, syntaxfel	
	12	Referensfel	Felaktig/saknad referens inom eller mellan dokument
	13	Standard	Följer ej standard
	14	Relevans	Innehåll ej relevant
	15	Redundans	
	16	Avsaknad	Dokument ej fullständigt
	17	Inkonsistens	Motsägelser
	18	Begriplighet	Formulering/jod onödigt svår att förstå
	19	Entydighet	Formulering mångtydig
SRS	21	Inkorrekt krav	
	22	Spårbarhetsproblem	
	25	Redundant krav	
	26	Saknat krav	
	27	Inkonsistent krav	
	28	Organisationsproblem	
	29	Entydighet	
	31	Verifieringsproblem	
SVVS,	41	Inkorrekt test	
SVVI &	42	Spårbarhetsproblem	
SVVR	45	Redundant test	
	46	Saknat test	
	48	Organisation	
	50	Respons	Förväntat resultat saknas/felaktigt i testinstruktion
	52	Testförutsättningar	Systemläge vid testfallets start saknas/felaktigt
	53	Testavslutning	Systemläge vid testfallets slut saknas/felaktig
STLDD &	61	Struktur	Indelning i moduler/delar olämplig
SDDD	62	Gränssnitt	
	63	Namngivning	olämplig/[följer ej regler]
	71	Realtidsproblem	kapplöpning/dödläge/handskakning/synkronisering
	72	Logiskt fel	
	73	Datafel	
	74	Timerproblem	
	75	Effektivitetsproblem	
Övrigt	100	Övrigt	

Bilaga A

Blanketter

Här finns pappersvarianter av följande blanketter:

- Statusrapport (1 sid)
- Granskningsprotokoll (2 sid)
- Testprotokoll (2 sid)
- Problemrapport (2 sid)
- Tidrapport (1 sid)
- Veckorapport (1 sid)

Dessa blanketter kan även laddas ner i pdf-format från kursens hemsida. Man kan antingen använda dessa blanketter eller motsvarande funktionalitet i det web-baserade stödsystem (“ePUSS” eller liknande system).

STATUSRAPPORT

Dokument: ______________________ Dokumentansvarig: ___________________________________

Förändringskontrollansvarig: ___________________________________

DEL A: Dokumentstatus

PR nr.	Åtgärdas ja/nej	Ansvarig för åtgärd namn	Deadline datum	Version nr. efter ändring	Godkänd datum	Kommentar

GRANSKNINGSPROTOKOLL

Granskningsdokument: _____ Datum: _____

Version: _____ Granskningsbeteckning: _____

DEL A: Totalt antal fel: A [ ] B [ ] C [ ]

DEL B: Granskningstyp

Granskningstyp: ❒ Formell ❒ Informell	❒ Ordinarie ❒ Omgranskning av _____

DEL C: Deltagare

Roll	Namn
Granskare	
Moderator	
Sekreterare	
Författare	

DEL D: Underskrift & beslut

Omgranskning ☐ Datum [ ]

Åtgärdas, därefter godkänd ☐ Senast [ ]

Godkänd utan åtgärd ☐

Protokolljustering	Underskrift för godkännade

DEL E: Granskningsanmärkningar

Löpnr.	Position	Feltyp	Grad	Beskrivning	PR nr.

GRANSKNINGSPROTOKOLL

Löpnr.	Position	Feltyp	Grad	Beskrivning	PR nr.

TESTPROTOKOLL

Specifikationsnummer: _____ Datum: _____

Specifikationsversion: _____ TP nummer: _____

DEL A: Totalt antal fel: A [ ] B [ ] C [ ]

DEL B: Testinformation

Testtyp: ☐ Funktionstest ☐ Systemtest ☐ Regressionstest

Testmiljö: ☐ Utveckling ☐ Drift

DEL C: Feltyper

Nummer	Feltyp	Grad	Betydelse
11-100	Se Projekthandledningen...	A	Omöjliggör fortsatt exekvering.
		B	Hindrar bara den testade funktionen. Exekveringen kan fortsätta.
		C	Funktionen kunde utföras men med smärre problem.

DEL D: Testresultat

Testfallsnr.	Korrekt ja/nej	Position i testinstruktion	Feltyp	Grad	PR nr.

TESTPROTOKOLL

Testfallsnr.	Korrekt ja/nej	Position i testinstruktion	Feltyp	Grad	PR nr.

DEL E: Underskrift

Test utfört av

PROBLEMRAPPORT

PR nummer: _____ Upprättad av (namn): _____

Upprättad (datum): _____

DEL A: Ursprung

❒ Granskning : granskningsbeteckning och löpnr: _____ Feltyp ☐

❒ Test : testfallsnr. & position: _____ Grad ☐

❒ Övrigt

Problemobjekt: _____ Version: _____

Position: _____

Problembeskrivning

DEL B: Utredning & beslut

Mottaget av FKG (namn): _____ Mottaget (datum): _____

Förkastas: ❒ Nej ❒ Ja

Anledning om förkastas: _____

Beskrivning av åtgärdsförslaget

PROBLEMRAPPORT

Dokumentpåverkan

Dokument	Version	Påverkan

Problemet infört i fas (nr.): ☐ Åtgärdas: Ja ❒ Nej ❒

Tidsuppskattning för åtgärd (min.): ☐ Ansvarig för åtgärd (namn): _____

Deadline (datum): _____

Anledning om ej åtgärdas: _____

DEL C: Åtgärd

Mottaget av åtgärdsansvarig(namn): _____ Mottaget (datum): _____

Tidsåtgång för åtgärd (min.): ☐

Beskrivning av utförd åtgärd

DEL D: Uppföljning & avslut

Åtgärden kontrollerad och godkänd av FKG (namn): _____

Datum: _____

Fas(nr.): ☐

TIDRAPPORT

Namn: ______________________________ Startdatum: __________

Projektgrupp: ______________________________ Vecka: __________

DEL A: Total effektiv arbetstid: [min] = [h min]

DEL B: Tidlogg för aktiviteter

Datum	Aktiv. nr.	Typ	Start	Stopp	Avbrott	Differens	Beskrivning

VECKORAPPORT

Namn: _____ Datum: _____

Projektgrupp: _____ Vecka: _____

DEL A: Total arbetstid denna vecka: min = h min

DEL B: Antal minuter per aktivitet

(summan av den totala tiden skall överföras till del A)

Nummer	Aktivitet	Typ				Total tid (min.)
		U	I	F	O	
11	SDP					
12	SRS					
13	SVVS					
14	STLDD					
15	SVVI					
16	SDDD					
17	SVVR					
18	SSD					
19	Slutrapport					
Summa						
21	Funktionstest					
22	Systemtest					
23	Regerssionstest					
30	Möte					
41	Föreläsning					
42	Övning					
43	Terminalövning					
44	Självstudier					
100	Övrigt					

DEL C: Tidsåtgång till olika aktivitetstyper

(värdena skall ej adderas till del A, skall redan var inkluderade)

Aktivitetstyp	Kod	Beskrivning	Summa
Utveckling och dokumentation	U	Skapande av ny kod, testfall och dokumentation inkluderat dokumentation av systemet.	
Informell granskning	I	Förberedelsetid och mötestid för informell granskning.	
Formell granskning	F	Förberedelsetid och mötestid för formell granskning.	
Omarbetning, förbättring och felreparation	O	Förbättring av effektiviteten eller tydligheten hos design samt reparation och ändring av fel i dokument och designobjekt.	

DEL D: Underskrift

Underskrift	Underskrift av ansvarig

Bilaga B

E-PUSS User Manual

E-PUSS User Manual

Computer Science, Lund University

1. Introduction

E-PUSS is an electronic reporting tool for the course in Large-Scale Software Development. (PUSS is the acronym for the Swedish name of the course.) The tools consist of three different parts: Time-, Problem- and Status reporting. *E*-PUSS is web based (see course web page for URL).

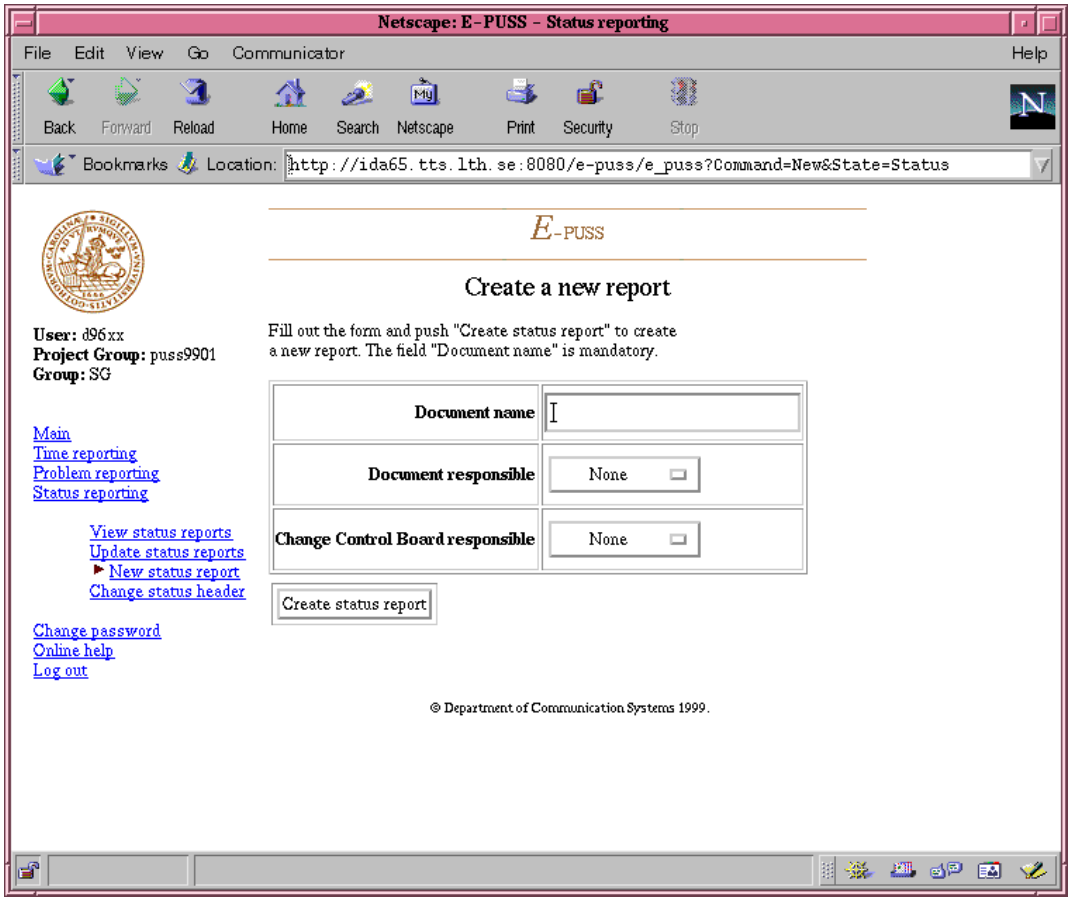
Knowledge about the different reports and how they are handled, is assumed in this manual. The different reports are:

- Week Report – used by all project members to record time spent on different activities.
- Problem Report – used to handle changes to baselined documents.
- Status Report – used to track the status of a document (configuration item).

Some help on the reports is available on the course web. Further information about the software process and the reports and documents can be found in the project manual.

The user manual is organized as follows. This chapter describes the general usage of *E*-PUSS. There are four subsequent chapters describing how to log in, how to use the time reporting, how to use the problem reporting and how to use the status reporting. Each chapter has several step-by-step instructions on how to use the different functions within *E*-PUSS. This is not a introduction to the different reports. This can be read in the project manual, which also describes how the different reports are supposed to be used.

1.1 General about *E*-PUSS



Navigation is done by using the menu, on the left. Click on the appropriate link to use a specific function. Information on what to do next will be presented in the main area. You can see where you are by which sub-menus that are present and further by which sub-menu that is marked with a red arrow.

Your user identification, project group and group within the project group is displayed in the top of the menu area.

1.1.1 When does changes take place?

Generally, no changes are done until you push a submit button. When you have submitted your changes, creation of new entries or updates of old ones, you will get a confirmation of the submittal. If you do not want submit a change, simply do not push the button to submit your update. Instead choose something from the menu, located at the left.

1.1.2 About the on-line help

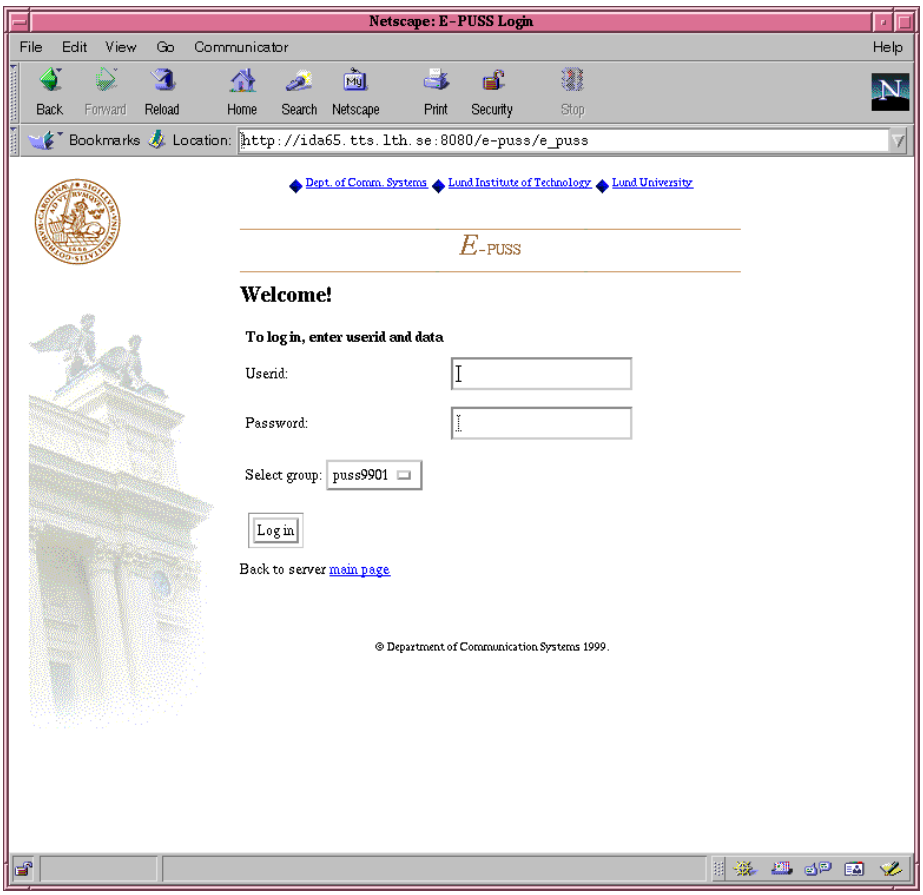
Some help is available on-line, including this user manual. When you use the on-line help, it is displayed in a separate browser window. Because of this, you can both use *E*-PUSS at the same time as you look at the help. Even though it is in another window, the on-line help still uses the same login as the ordinary *E*-PUSS window. That is, you cannot view the help if you are not logged in to *E*-PUSS.

2. Log in and out, changing password and organizing group

Authentication for *E*-PUSS depends on cookies[1]. Make sure your web browser is set to accept cookies. By default most browsers accept cookies.

2.1 Login in

To log in to *E*-PUSS, point your web browser to the URL given by the department. The first page you will see is the login page. Enter your userid and password in the corresponding fields. Also, you need to select your project group from the menu. Click “Log in” to open a connection to the *E*-PUSS system.



A login is not allowed to be passive for more than 15 minutes. If a connection is not used in that time then the connection is automatically closed and the user is logged out. This is to prevent someone else from using your login, if you forget to log out.

If you are logged out due to the time-out described while editing something in *E*-PUSS, there is a auto recovery function. *E*-PUSS will prompt you as you try to submit a change that you have been logged out because the inactivity exceeded 15 minutes and provide a login form. The command or change that could not be performed will be executed when you log in again and you will be prompted with the result as usual.

1. A cookie is a piece of information which is set on the client side and used by the web server to store session information.

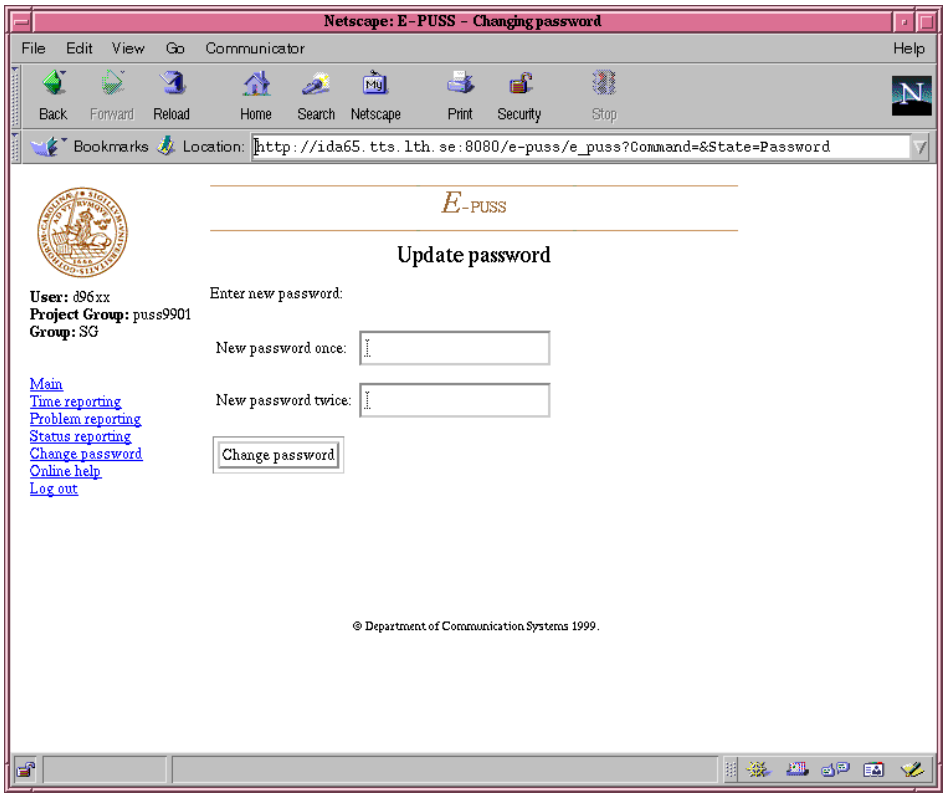
2.2 Login out

To log out simply click on the link “Log out” in the menu on the left. The system will confirm that you have logged out.

It is important that you always logs out when you have finished working with *E*-PUSS. It is only allowed to be logged in simultaneous once per user. If the systems detects a second attempt to log in from one user at the same time the second attempt is prevented and the login attempt will fail. If you for some reason is not allowed to access *E*-PUSS due to simultaneous logins, wait for 15 minutes for the first login to be automatically logged out and then try to log in again. Note that the auto recovery function does not apply to the situation described here.

2.3 Changing password

You can change the password by choosing “Change password” from the menu. Enter the password twice and then click the submit button to submit the updated password.



Note: You should change the password given to you at registration to *E*-PUSS.

2.4 Organize group - restricted to project managers

If you belong to the project management group, then you are able to organize the group. By clicking on “Main” and then choosing “Organize group” you can adjust the group of the project group members. There are nine categories to choose from, see table 1. There is a tenth group category, namely CUS, for **Cus**tomer. However, only Customers and System administrators have access to change users group to and from this category.

TABLE 1. Available groups

Category	Group	CCB
PG	**P**roject Management **G**roup	Yes
SG	**S**ystem Management **G**roup	Yes
TG1	**T**est **G**roup **1**	No
TG2	**T**est **G**roup **2**	No
DGCH	**D**esign **G**roup **Charging**	No
DGTC	**D**esign **G**roup **T**ake **C**all	No
DGMA	**D**esign **G**roup **M**aintenance	No
DGCFU	**D**esign **G**roup **C**all **F**orward **U**nconditional	No
DGCFN	**D**esign **G**roup **C**all **F**orward **N**o Answer	No

Note that the **C**hange **C**ontrol **B**oard, CCB[1], consists of PG and SG. Since the members of CCB have special privileges, it is important that PG at an early stage in the project organizes the group. Otherwise, PG will be the only member of CCB.

3. Time reporting

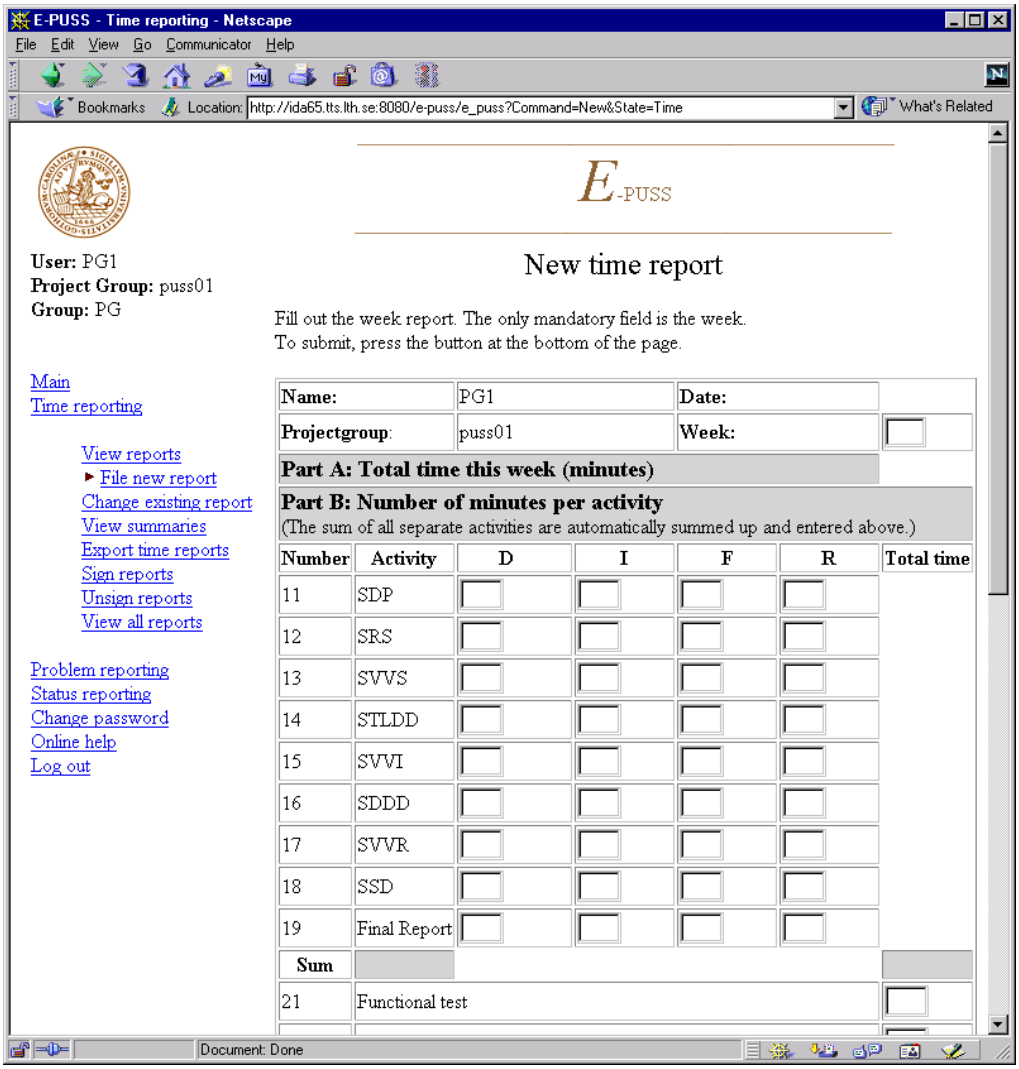
Metrics of time spent on different activities in the project are collected throughout the project. The data collected should be handed in once a week, using *E*-PUSS. When the reports have been handed in, PG signs them electronically.

All the group members have to keep track of their time on their own. You have got a paper form to keep track of the activities during the week. The time reports submitted through *E*-PUSS is a summary of the time spent each week, i.e. a week report.

3.1 Filing out a new week report

1. If you are not logged in to *E*-PUSS, log in now.
2. Select “Time reporting” from the menu.
3. Select “File new report”. Your user id, project group and the current date is automatically entered.
4. Enter the time spent, in minutes, on each activity for the week you are writing a report. The activities of which you have not spent any time on is to be left blank. When you have submitted the report, *E*-PUSS will add up the different sums present in the report.
5. When you are ready, push the button at the bottom of the page to submit the week report. You will now see what you have entered.

1. FKG is the Swedish acronym.



3.2 Changing an existing report

1. If you are not logged in to *E*-PUSS, log in now.
2. Select “Time reporting” from the menu.
3. Select “Change existing report”. All your time reports will now be summarized on one page. Select, using the radio buttons to the left, which report you want to update. Push “Get week” to get the report.
4. Do the necessary updates. If you decide not to update a report, simply make a selection from the menu to abort the updates. No change is done unless you push the submit button.
5. When you are done, submit the changes by pushing the button at the bottom of the page. When you have submitted your changes you will see the altered week report.

If a report has been signed by a project manager then you are not allowed to update the report. To be able to update a signed report you must notify someone in PG to unsign it. Only after that is an update possible. *Note* that your project managers have to sign the report again when you are done.

3.3 View week reports

1. If you are not logged in to *E*-PUSS, log in now.
2. Select “Time reporting” from the menu.

3. Select “View reports”. All your time reports will now be summarized on one page. Select, using the radio buttons on the left, which report you want to view. Push the button at the bottom to get the report.

You can only see your own time reports. For project managers, who needs to see the time reports from all project group members, see below.

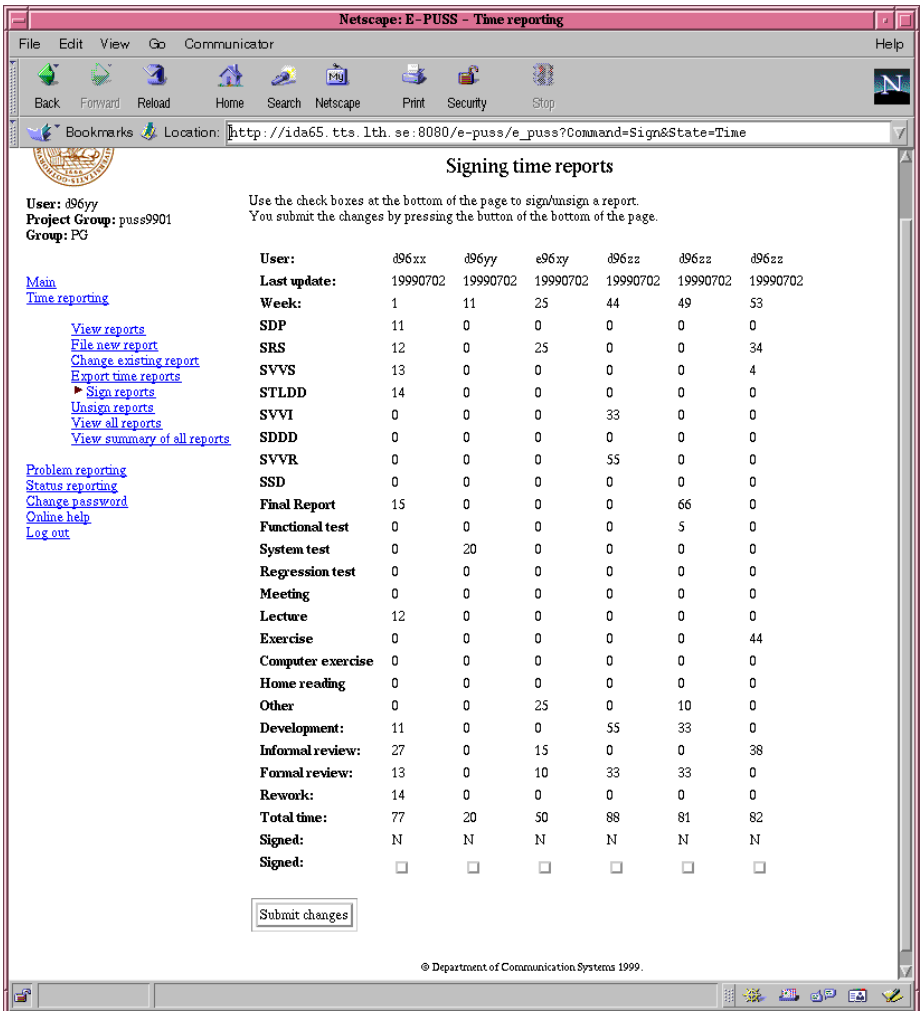
3.4 View summaries

To view the summaries for a week, select “View summaries” in the time reporting meny. Then select a week and you will get the summary of each activity spent this week. Viewing summaries are similar to viewing a single report. You can also select all weeks, that is a summary on all activities for the entire project.

3.5 Signing and unsigning reports - restricted to project managers

Project management Group is supposed to sign all week reports handed in. Do the following to sign reports:

1. If you are not logged in to *E*-PUSS, log in now.
2. Select “Time reporting” from the menu.
3. Select “Sign reports” from the menu. A summary of all the unsigned reports is shown. (To investigate each report, choose “View all reports”, see later.)
4. Click the boxes at the bottom to sign the reports.
5. Hit the update button to submit changes.



If someone wants to change a signed report, a member from PG must first unsign it to allow the change to take place. Unsigning is done in a similar way as signing is, but select “Unsign reports” from the menu, instead of “Sign reports” in step 3 in the instructions above. Also, the sign boxes is located to the left, instead of at the bottom. Note that when a project group member has updated a report you must sign it again.

A project manager does not have the privileges to update week reports for others than themselves.

3.6 View all reports - restricted to project managers

For a project manager there is the possibility to view reports of all project members.

To view the week reports separately, click in “View all reports”. The selection of which report to further investigate is done similar to viewing your own reports. In the overview, the reports are primarily sorted by week, secondarily by user id.

3.7 Exporting the week reports - restricted to project managers

To be able to use the data in Matlab, Excel, or some other program, project managers have a possibility to export all the time data from the group. This is done by selecting “Export time reports” from the time reporting menu. The format is a tab separated list of all the data in the system. Then you can choose to export either just the reports for a specific week or all the reports for the project.

The activities are given by their number. For documents a letter is also given. This letter is either D, I, F or R. These stand for **D**evelopment, **I**nformal Review, **F**ormal Review and **R**ework. The activity numbers and type are specified in the project handbook.

4. Problem reporting

The handling of problem reports in the course is also done through *E*-PUSS. The tool supports the problem reporting process used in the TUP[1]. *E*-PUSS emphasizes on awareness. To facilitate this, there is only a limit number of things done automatically. Hence, this is considered a feature. For example, there is made no automatic update facility for the status reports when a problem report is submitted.

4.1 Introduction to problem reporting in *E*-PUSS

A problem report in *E*-PUSS has one of the following states:

- *Created* - A problem report has been created and the source of the problem will be investigated. Everyone is allowed to change the report in this state.
- *Investigation* - Someone from CCB (Change Control Board) has noted that the problem report exist and is investigating the problem. Only members of CCB is allowed to update the report.
- *Customer* - From the investigation performed by CCB it was concluded that the customer should see this report and approve the actions suggested by CCB. CCB is allowed to make updates on the reports in this state.
- *Fixing* - The problem at hand should be fixed and this is in progress. All group members have access to make changes on the report.
- *Rejected* - From the investigation it was decided that this problem report should not be dealt with and thereby it is rejected. Only CCB is allowed to make changes in this state.
- *Fixed* - The problem reported is fixed and the fix is checked and approved by CCB. Only CCB is allowed to do additional changes to the report.

Only members of CCB is allowed to update the state of a report, see below. In all states everyone is allowed to view the problem reports.

4.2 Creating problem reports

Problem reports are created as follows:

1. If you are not logged in to *E*-PUSS, log in now.
2. Select “Problem reporting” from the menu.
3. Select “New problem report”.
4. Fill out the appropriate fields. Some fields are filled out for you. If those values are not correct they can be revised. The fields in the header of the report are the only mandatory fields to fill out. These must be filled out.
5. When done with writing the report click the submit button to submit your new report.

When a report is created it’s state is automatically set to *Created*.

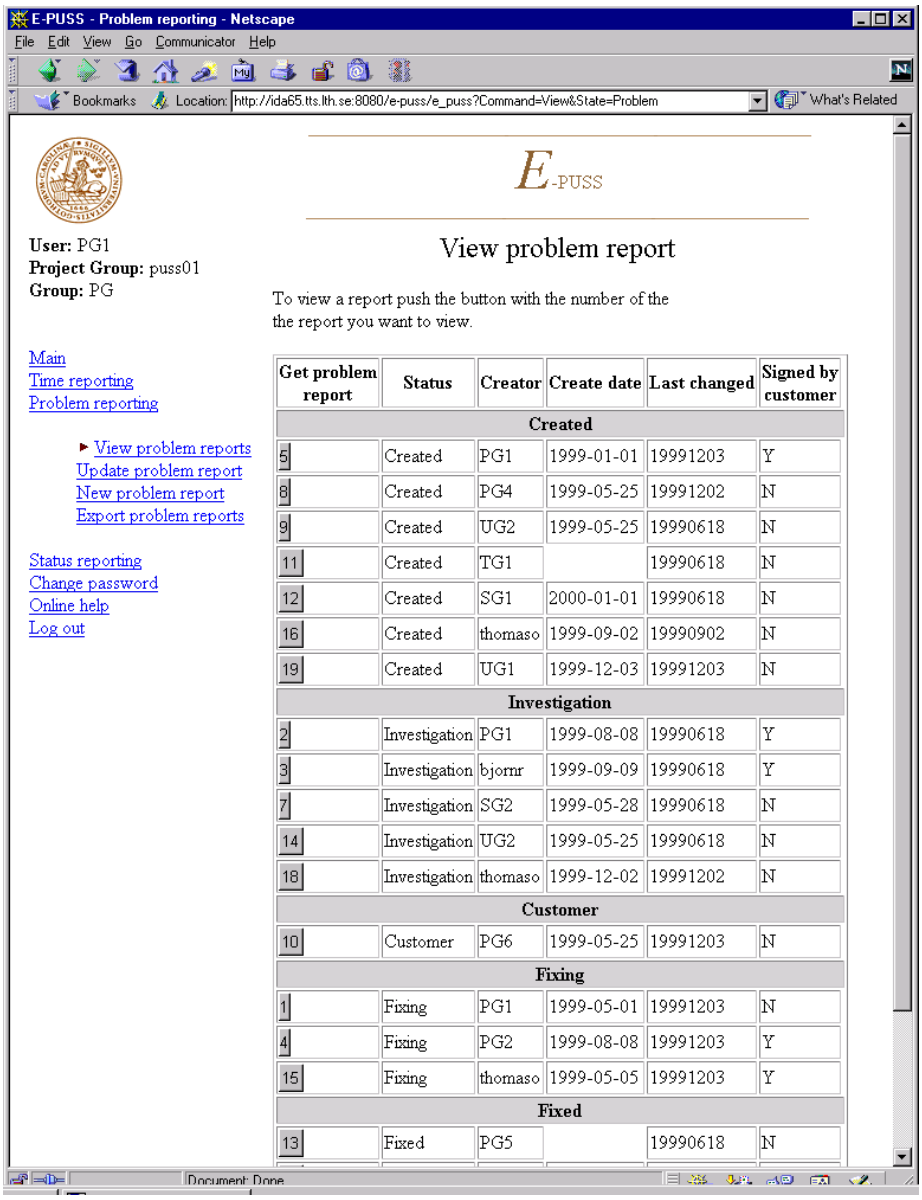
1. Telecoms Utvecklingsmodell för Programvara

4.3 Viewing problem reports

By clicking on “View problem reports” in the problem menu you will get an overview of all the problem reports for the project group. They are sorted by state, in logical order. *Fixed* and *Rejected* reports are at the end and *Created* at the beginning.

1. If you are not logged in to *E*-PUSS, log in now.
2. Select “Problem reporting” from the menu.
3. Select “View problem report”. You will se an overview of all the reports, sorted by status and problem report number.
4. Select a problem report to update by pushing its number under “Get problem report”.

When you have selected a report you will see a button at the bottom of the page, after the report. If you press this button you will change to update mode for the report (see below). This is a quick hand to ease the use of the *E*-PUSS problem reporting tool.



Get problem report	Status	Creator	Create date	Last changed	Signed by customer
Created					
5	Created	PG1	1999-01-01	19991203	Y
8	Created	PG4	1999-05-25	19991202	N
9	Created	UG2	1999-05-25	19990618	N
11	Created	TG1		19990618	N
12	Created	SG1	2000-01-01	19990618	N
16	Created	thomaso	1999-09-02	19990902	N
19	Created	UG1	1999-12-03	19991203	N
Investigation					
2	Investigation	PG1	1999-08-08	19990618	Y
3	Investigation	bjornr	1999-09-09	19990618	Y
7	Investigation	SG2	1999-05-28	19990618	N
14	Investigation	UG2	1999-05-25	19990618	N
18	Investigation	thomaso	1999-12-02	19991202	N
Customer					
10	Customer	PG6	1999-05-25	19991203	N
Fixing					
1	Fixing	PG1	1999-05-01	19991203	N
4	Fixing	PG2	1999-08-08	19991203	Y
15	Fixing	thomaso	1999-05-05	19991203	Y
Fixed					
13	Fixed	PG5		19990618	N

4.4 Updating problem reports

Updates on reports with status *Created* and *Fixing* is open to all project group members. For a status different than those, updates is restricted to CCB. A report is selected for updates the same way it is selected for viewing, see above.

Only one of the four parts can be updated at a time. Part A is updated when the status is *Created*, part B is updated when the status is *Investigation*, part C when status is *Fixing* and part D when status is *Fixed*. When a report has the status *Rejected* no parts of the report can be updated. In state *Customer* it is possible to update part B. Also, the header of the report can be updated in state *Created*.

When you update a report, *E*-PUSS will automatically let you update the correct part. Which part this is can be seen in the introduction to the problem reports, see above. The user never has to be concerned with which part to update.

1. If you are not logged in to *E*-PUSS, log in now.
2. Select “Problem reporting” from the menu.
3. Select “Update problem report”. You will se an overview of all the reports, sorted by status and problem report number.
4. Select a problem report to update by pushing its number under “Get problem report”.
5. *E*-PUSS automatically lets you update the appropriate part of the report. The only mandatory fields is the ones in the header of the report. These can only be updated when a problem report is in state *Created*.
6. When done push the “Update button” at the end of the page to submit changes. The report will now be updated.

If you typed an error or if you want to alter anything in the report you can always go back and change the report again.

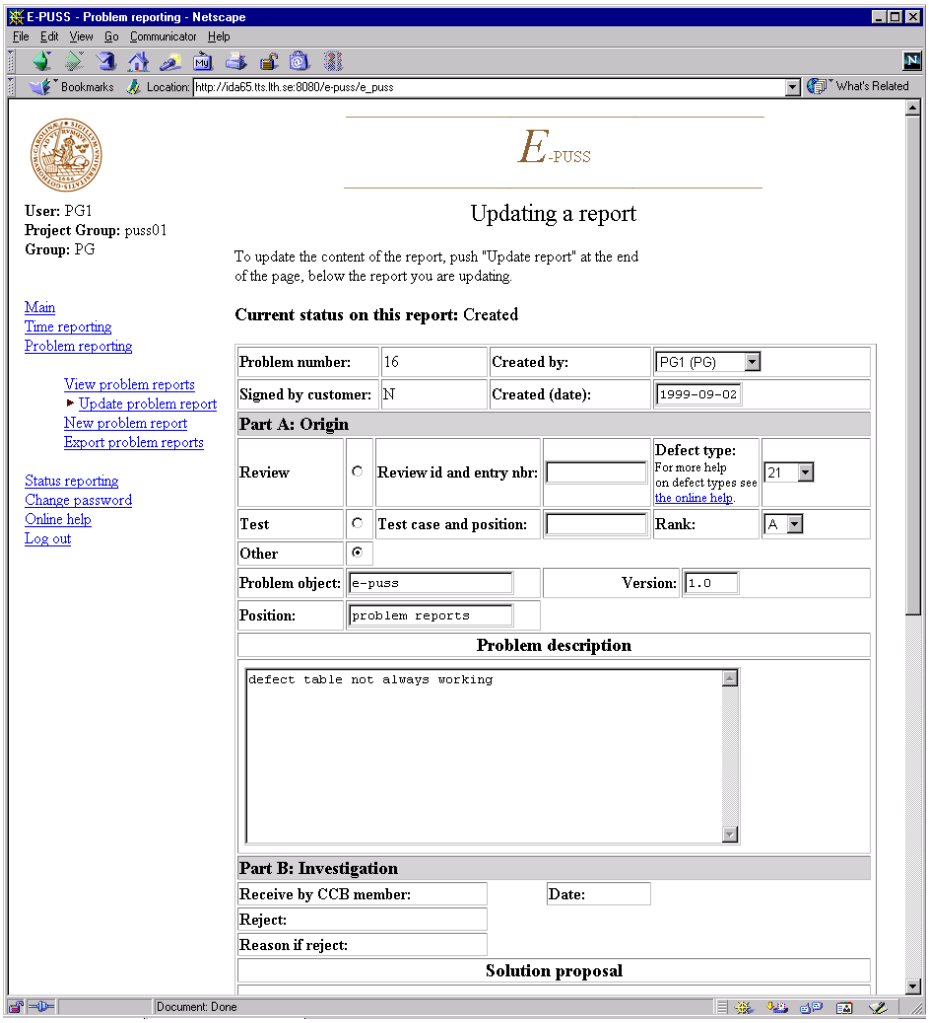
4.5 Changing state of a problem report - restricted to CCB

As described in the project handbook a problem report follows a certain flow. The different states of a report represents this flow. To update the status of a problem report you have to be a member of the Change Control Group.

1. If you are not logged in to *E*-PUSS, log in now.
2. Select “Problem reporting” from the menu.
3. Select “Update problem report”. You will see an overview of all the reports, sorted by status and problem report number.
4. Select a problem report to update by pushing its number under “Get problem report”.
5. If you are a part of CCB then *E*-PUSS will display buttons at the bottom of the page. Which buttons that appear depends on which state the report currently is in.
6. By clicking one of the buttons the status of the report is changed. Nothing else on the report is changed, only the state. If you have done updates in the report which you have not submitted then these are lost, as for all other changes in the report.

4.6 Exporting problem reports - restricted to project managers

It is possible to export the problem reports. The feature does a test dump of all the report in a tab separated list for the project group. The problem reports are exported by selecting “Export problem reports” from the meny. This feature has no further support from the department.

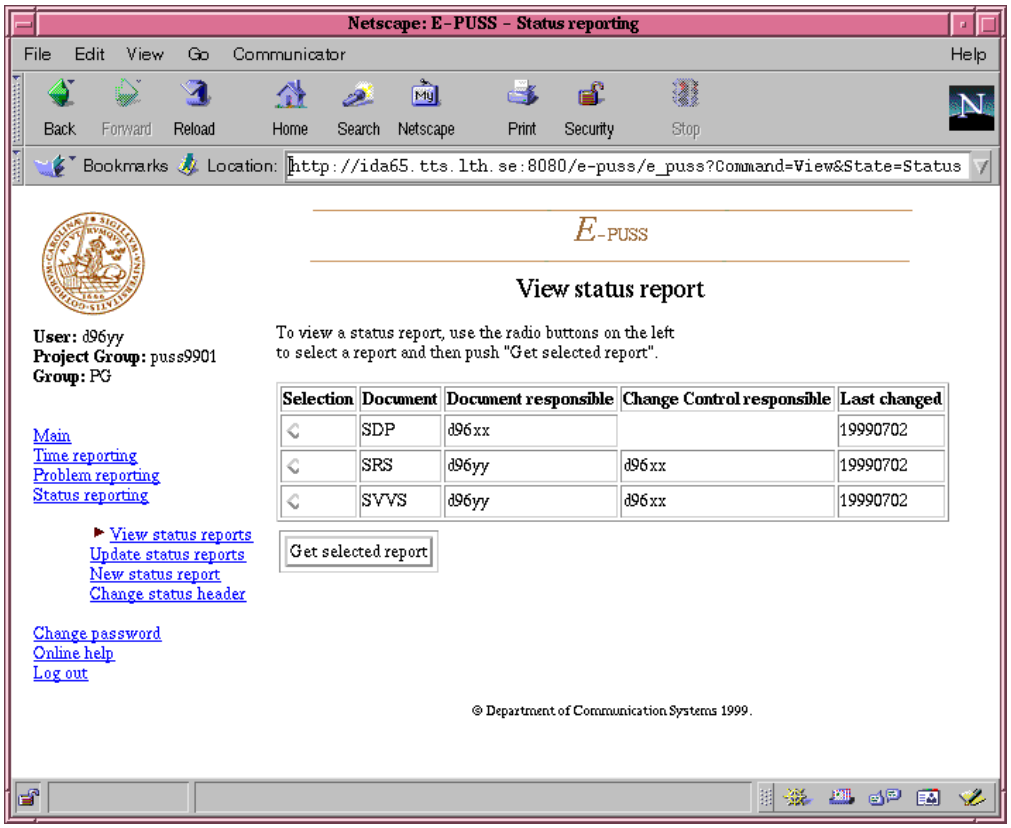


5. Status reporting

A status report keeps track of all the changes in the documents in the project. Only CCB is allowed to update and to create new status reports. Everyone is allowed to view the status reports within the project group.

5.1 Viewing status reports

1. If you are not logged in to *E*-PUSS, log in now.
2. Select “Status reporting” from the menu.
3. Select “View status report”. You will see the header information of each status report for the group, and also the date of the last update of each report.
4. Select a status report to update by clicking on one of the radio buttons. Get the report of your choice by pushing the button at the bottom of the page.



5.2 Creating new reports/changing header data - restricted to CCB

The only mandatory field on a status report is the document name in the header. This field must be filled out. Creating new status reports or changing header data on one is restricted to CCB members.

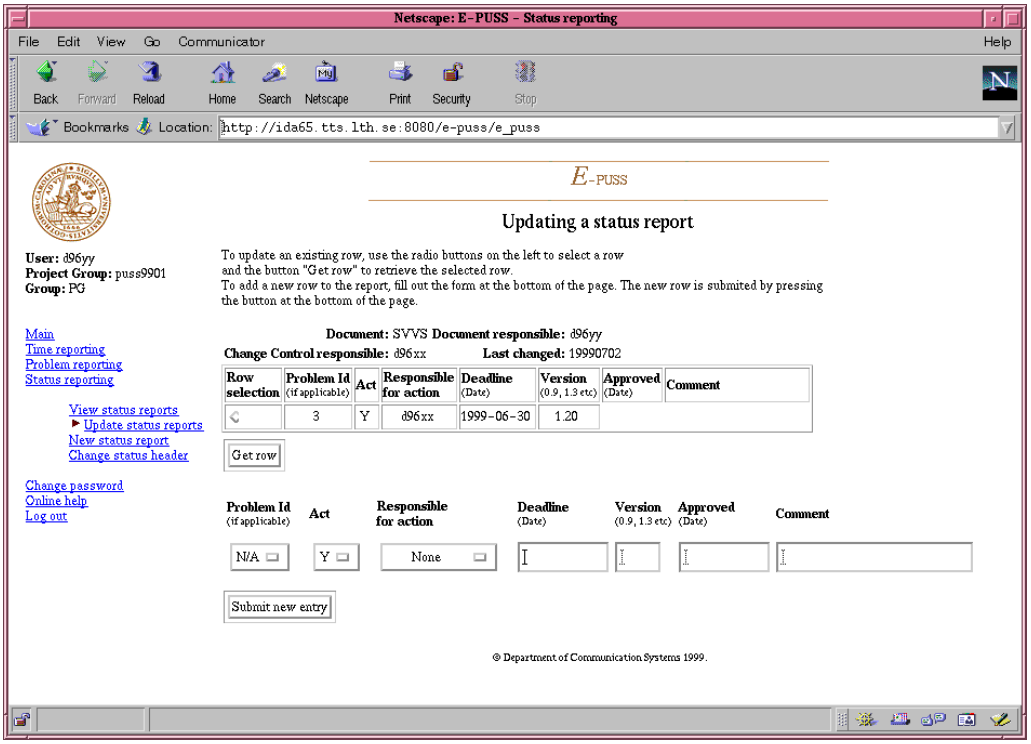
To create a new status report select “New status report” in the status menu. Fill out the document name and use the pull down menus to choose document and CCB responsible. You create the new status report by pushing the button “Create status report”.

To change the header information on a status report, select “Change status header” on the status menu. As when selecting a report to view, you select the status report to update using radio buttons. You get the report by pushing the button after the table.

5.3 Update status report - restricted to CCB

All changes on a status report is restricted to CCB members only.

1. If you are not logged in to E-PUSS, log in now.
2. Select “Status reporting” from the menu.
3. Select “Update status report”. You will see the header information of each status report for the group, and also the date of the last update of each report.
4. Select a status report to update by clicking on one of the radio buttons. Get the report of your choice by pushing the button at the bottom of the page.



When you have selected a status report there are two ways of updating it. Either you create a new row:

- If you have done step 1 to 4 in the instructions above, then there are forms at the bottom of the page. Fill these out with the new information.
- Submit the new row by pushing the button “Submit new entry”.

Or you can update an existing row:

- Select a row to update, marking your selection with the radio button. Get the row by clicking on the button “Get row”.
- Update the row with the changes you have.
- Submit the changes for the row. This is done by pushing “Submit change”.

5.4 Export status reports - restricted to project managers

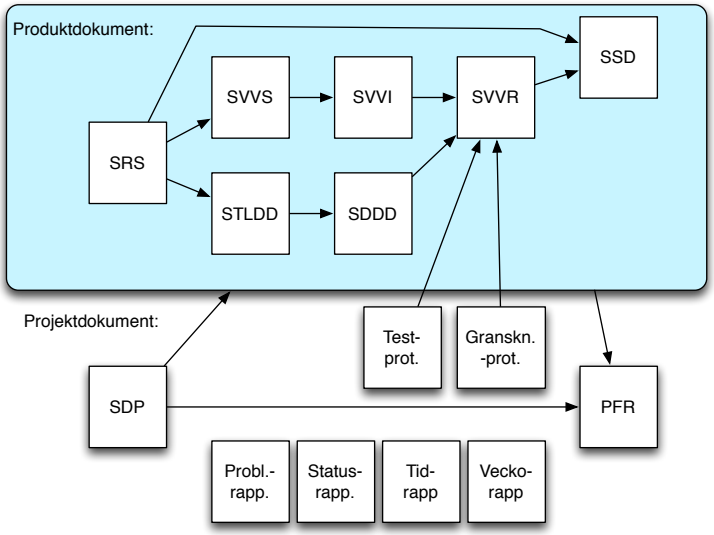
This is done by selecting “Export status reports” from the status reporting menu. The format is a tab separated list of all the status reports data in the system. This feature is not further supported by the department.

	Nr	Feltyp	Eventuell beskrivning
Generella	11	Stavfel, syntaxfel	
	12	Referensfel	Felaktig/saknad referens inom eller mellan dokument
	13	Standard	Följer ej standard
	14	Relevans	Innehåll ej relevant
	15	Redundans	
	16	Avsaknad	Dokument ej fullständigt
	17	Inkonsistens	Motsägelser
	18	Begriplighet	Formulering/jod onödigt svår att förstå
	19	Entydighet	Formulering mångtydig
SRS	21	Inkorrekt krav	
	22	Spårbarhetsproblem	
	25	Redundant krav	
	26	Saknat krav	
	27	Inkonsistent krav	
	28	Organisationsproblem	
	29	Entydighet	
	31	Verifieringsproblem	
SVVS,	41	Inkorrekt test	
SVVI &	42	Spårbarhetsproblem	
SVVR	45	Redundant test	
	46	Saknat test	
	48	Organisation	
	50	Respons	Förväntat resultat saknas/felaktigt i testinstruktion
	52	Testförutsättningar	Systemläge vid testfallets start saknas/felaktigt
	53	Testavslutning	Systemläge vid testfallets slut saknas/felaktig
STLDD &	61	Struktur	Indelning i moduler/delar olämplig
SDDD	62	Gränssnitt	
	63	Namngivning	olämplig/[följer ej regler]
	71	Realtidsproblem	kapplöpning/dödläge/handskakning/synkronisering
	72	Logiskt fel	
	73	Datafel	
	74	Timerproblem	
	75	Effektivitetsproblem	
Övrigt	100	Övrigt	

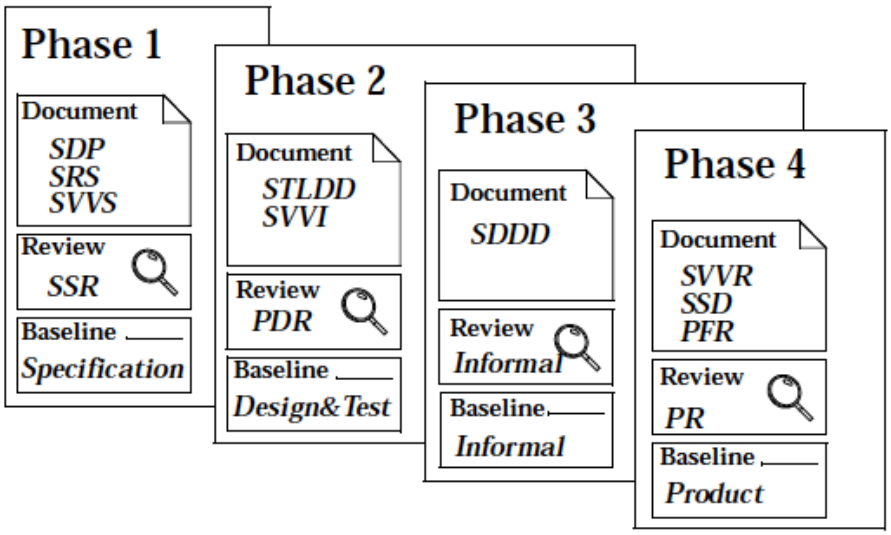
Felklasser

A (Allvarligt fel): Felet omöjliggör fortsatt exekvering. Dessa fel ska alltid korrigeras.
B (Mindre fel): Felet hindrar bara den testade funktionen. Exekveringen kan fortsätta. Dessa fel ska alltid korrigeras.
C (Obetydligt fel): Funktionen kunde utföras men med smärre problem.

Allvarlighet fÃ¶r fel



Dokument och relationer mellan dokument



Utvecklingsmodell