Dokument

Detta är en liten vägledning för att skriva och bedöma dokument i kursen TDDD96.

En fungerande produkt är mer än bara kod. Det tillkommer dokumentation, juridiska dokument, utbildning, underhåll och service.

I kursen krävs ett antal dokument, vissa av dem lämnas in flera gånger för meningen är att de skall hållas aktuella och förbättras. Ett exempel på detta är en projektplan. Andra dokument skriver ni enligt överenskommelse med kund, exempelvis användarhandledning, dem tittar vi gärna också på ifall vi har någon feed-back.

Anledningen till att vi har en hel del dokument i kursen är inte för att vi älskar dokument eller inte är medvetna om moderna arbetssätt. Dokumenten kommer att hjälpa er i projektet och genom att skriva och uppdatera dokumenten kommer ni tvingas att vara tydliga i er tankeprocess.

Ett mål med kursen är att skriva ett kandidatarbete och dokumentskrivandet är mängdträning i att skriva begripligt. Dokumenten används också för examination – för att följa projektets status vid sidan av handledarmöten, seminarier och demonstrationer.

Se till att dokumenten är användbara för er. Skriv till exempel inte en planering som ni inte tänker följa bara för att få till ett snyggt dokument som överensstämmer med en standard. Se till att ni håller dokumenten uppdaterade så får ni glädje av dem-

Dokument i kursen

- Projektplan
- Kravspecifikation
- Kvalitetsplan
- Arkitekturdokument
- Testplan
- Testrapport – sedan 2022 ingår i veckorapporter

Allmänt

Dokumentet är daterat.

Dokumentet har ett versionsnummer.

Det finns en ändringshistorik, med kortfattad men konkret information om det viktigaste som skett. Det duger inte att skriva ”Kalle fixade till en del nya saker”. Skriv istället ”Specifikation för server-delen tillagd.” Man kan ha en kolumn med ansvarig, där Kalle kan skriva sitt namn.

Stega inte versionsnummer för småsaker, detaljerna har ni i Git.

Projektet är identifierat.

Det finns kontaktuppgift till någon som kan lämna mer information.

Nödvändiga referenser till källor för att förstå dokumentet finns med.

Språket är begripligt, något enstaka skrivfel per sida är OK i inlämning 1.

Syftet med dokumentet är angivet.

Det finns en beskrivning av tänkta läsare av dokumentet.

Projektplan

Projektplanen har rubriker enligt PUM-teorin:

Utbildning är vanligtvis en stor del av projekten eftersom det är en kurs. Det är helt OK att göra en egen utbildningsplan, som lämnas in tillsammans med övriga dokument

3-10 risker, helst projektspecifika och någon teknisk risk

Detaljnivån avtar med avståndet i tid. För inlämning 1 kan planeringen av Vt2 är endast översiktlig. Är planerna för maj mycket detaljerade bör man fråga hur realistiska de är. Däremot skall man veta mycket om vad som skall ske de närmaste veckorna.

Någonstans behöver man skriva om hur produkten skall levereras. Projektplanen kan vara ett bra ställe. Skulle leveransen vara tekniskt komplicerad, kan det vara bättre att skriva en egen leveransplan.

Projektplanen behöver ses över ofta, börja med en gång i veckan.

Kravspecifikation

Kurskrav: I den sista versionen skall det finnas tre mätbara kvalitetskrav på produkten som kan vara uppfyllt till olika grad, dvs vi accepterar inte formuleringar av mål på nominalskala. Handledaren rapporterar till examinator när det målet är uppfyllt.

Det finns någon form av översiktlig beskrivning av kraven på systemet. Exempelvis användningsfall. Motsvarar kap 2 i IEEE 830-std.

Individuella krav är numrerade och otvetydiga.

Textuella formuleringar av individuella krav är fullständiga meningar. Vaga modala verb som ”bör” eller ”önskar” skall undvikas.

Det finns någon form av klassificering av krav efter hur de kommer att behandlas i projektet. Exempelvis funktionella krav och icke-funktionella krav, som ju testas på helt olika sätt.

Det skall framgå vilka krav som är kundkrav och vilka som är tekniska krav framtagna av gruppen själv.

Kraven skall vara prioriterade, men det är knappast ändamålsenligt att ha fler än 2 prioriteringsnivåer. Om gruppen har fler kan man fråga sig varför. Troligtvis innebär prioritet 3 att kraven aldrig kommer att implementeras – så varför ha med dem?

Om kunden är angiven som målgrupp, framgår det om hela kravspecifikationen eller bara delar är skrivna för kunden. Dessa delar är det rimligt att antaga att kunden förstår. Pleeger och Atlee skriver om en kundspecifikation och en mer detaljerad teknisk specifikation. En detaljerad teknisk specifikation kan behövas om man skall lämna specifikationen vidare till någon annan för implementation. Det är inte fallet i studentprojekt.

Man bör sträva efter att ha krav på en hög abstraktionsnivå. I synnerhet vid första inlämningen. En tumregel kan vara att man bara har med saker som man måste prata med kunden om ifall de behöver ändras.

Om man har en lång lista av detaljerade krav kan man behöva lägga in kraven i något verktyg med databas så att man kan ha länkar, taggar och olika sökfunktioner. Då genererar man specifikationer efter de sökningar man gör och skriver inte upp dem i ett ändlöst textdokument. Det är knappast aktuellt för studentprojekten.

IEEE830:1998 ger en bra och lättläst vägledning om vad som kan vara med i en kravspecifikation. Det är speciellt sympatiskt att standarden uppmuntrar till att man anpassar sitt dokument till det system man har. Det finns till och med exempel på hur man kan göra olika dispositioner av de detaljerade kraven i kapitel 3.

https://doi.org/10.1109/IEEESTD.1998.88286

Kvalitetsplan

Vi har ett kurskrav på att den skall finnas en process i fokus. Den processen skall vara välbeskriven, man skall kunna redovisa hur man utvärderar den och hur den skall förbättras. All denna information finns inte till inlämning 1, men själva processen skall vara identifierad då. Handledaren rapporterar till examinator när det uppnåtts.

Det finns också ett kurskrav på att man skall ha ett mätprogram, dvs en plan över vilka mätningar man kontinuerligt skall genomföra på produkt, process och resurser.

Kvalitetsplanen skall innehålla information om processer för statisk kontroll av arbetsprodukter, exempelvis inspektioner eller källkodanalys.

Det finns information om andra processer som bidrar till kvalitet, exempelvis konfigurationshantering, ändringshantering, riskhantering.

Det finns också information om samarbete och kommunikation.

Det finns information om vem, eller vilken konstellation av projektmedlemmarna, som kan fatta beslut.

När det gäller processer är det mycket vanligt att man hellre beskriver dem i projektplanen, så det är OK med många referenser till den.

Det finns information om vem som ansvarar för att initiera processer och utvärdera dem.

Det finns information om hur man tänker att de processer man har bestämt skall utvärderas.

Det finns information om hur man kan förbättra både processer och produkten.

Det finns information om vilken typ av dokumentation som kommer att produceras.

Testning planeras i egen dokumentation, så den skall också vara refererad till. Det gör inget att testplanen är rudimentär vid inlämning 1.

En kvalitetsplan skall vara skriven så att den kan läsas och förstås av en utomstående, men kunnig person. Kunden skall idealt kunna låta en konsult gå igenom kvalitetsplanen och säga om den bedöms som tillräcklig.

Kvalitetsplanen läses av granskare för en certifiering, typ CMMI eller ISO 9000-familjen. Men det är ett för högt mål för ett studentprojekt.

IEEE730:2014 är inte helt lätt att läsa (2002 år version var faktiskt lättare). Kapitel 4 är som en liten lärobok i hur SQA fungerar i en organisation. Kapitel 5 beskriver hur man kan beskriva sina processer för SQA. De är indelade i olika aktiviteter: SQA processer, produktens kvalitet och utvecklingsprocessernas kvalitet. Figur 5 visar ett antal rubriker för en SQA plan, varav en hel del inte är tillämpliga för ett studentprojekt. Appendix C är en vägledning för vad man kan behöva tänka på när man tar fram en kvalitetsplan. Den består av en massa frågor som är bra att gå igenom för att förvissa sig att man inte glömt något, exempelvis kundens involvering.

https://doi.org/10.1109/IEEESTD.2014.6835311

Arkitekturdokument

Det finns någon form av systemöversikt som tar upp tilltänkta användare och de viktigaste kraven, inklusive kvalitetskrav och begränsningar på möjliga realiseringar av systemet. I OpenUP kallas dessa arkitekturellt signifikanta krav.

Det finns någon form av spårbarhet till kraven, exempelvis kravnummer.

Det finns både en beskrivning av systemets statiska struktur och en beskrivning av systemets dynamiska beteende.

Om det är viktigt, finns det flera vyer av systemet, exempelvis implementationsvy och exekveringsvy.

Det går att utläsa vilka de viktigaste komponenterna är och hur olika gränssnitt dem emellan ser ut.

Det finns någon form av motivering till valet av arkitektur.

Det finns någon form av beskrivning av lösningar som förkastats.

En riktigt bra arkitekturbeskrivning fungerar som en sorts anteckningsbok där man kan följa hur arkitekturellt signifikanta krav utvecklas från systemegenskap till faktisk lösning. Där följer

motiveringar till val och varför vissa lösningar förkastats som en naturlig del. För kunden kanske detta till och med kan vara ett viktigare dokument än koden.

Till inlämning 2 bör de ha en mycket bra arkitekturbeskrivning, som sedan fylls på under projektets gång.

OpenUP besrkiver en Architectural Notebook, som är en den bästa dokumentationshandledningen jag sätt. Där finns länkar till checklistor och mallar.

https://www.ida.liu.se/~TDDC88/openup/practice.tech.evolutionary_arch.base/workproducts/architecture_notebook_9BB92433.html?nodeId=9351a72b

Mer designdokumentation behövs nog inte för denna kurs. För större projekt och projekt där man skall lägga ut kodningen på någon annan behövs mer. Då beskriver man de enskilda komponenterna vad gäller struktur och beteende. Kritiska algoritmer specificeras. UML-diagram eller liknande diagram minskar risken för missförstånd. Det kan vara svårt att hålla designen uppdaterad och det finns programvara som kan generera diagram direkt från koden. Det finns dock inget som hindrar att man lägger in mer detaljerad information i arkitekturdokumentet om den är mycket viktig för projektets framgång, exempelvis specifikation av kritiska algoritmer.

Testplan

Det finns en översikt om vad som skall testas. En tredjepartskomponent kanske inte testas.

Det finns en översikt över vilka typer av tester som skall göras, exempelvis funktionstest och användbarhetstestning.

Det finns en översikt över vilka nivåer[1] av tester som skall göras, exempelvis enhetstest, integrationstest, systemtest

Det finns en spårbarhet mellan krav och tester, åtminstone på en hög nivå.

Testomgivningen är beskriven. Ibland kan utveckling och validering av testomgivningen vara en stor del av projektet. Då kan man diskutera med gruppen om testplanen räcker, eller om man måste behandla testomgivningen som ett eget projekt.

Det finns någon form av testprocess. Eventuella beroenden mellan tester är utredda.

Det finns någon plan för hur man skall hantera testfall som inte går igenom. Exempelvis kan man välja att fel i enhetstest rättas direkt av utvecklaren, medan större fel dokumenteras och följs upp.

Det framgår hur man samlar data till metriker, både vad gäller systemet och själva testningen.

Det finns en förteckning av testfallen med minst: tesfallsidentitet, indata och förväntade utdata. Ibland kan det också behövas en översikt över grupper av testfall.

Vi förutsätter att grupperna försöker automatisera testningen så långt det går. Det innebär att en hel del information hanteras av verktyg, typ JUnit. I så fall innehåller tesplanen referenser dit. Det finns ingen anledning att duplicera information

Explorativ testning har kommit i ropet igen. Det är fullt OK om grupperna använder detta mindre detaljplanerade, men mer utforskande sätt att testa, men då skall tiden man avsätter för detta vara

[1] Tyvärr använder IEEE-829 två betydelser av “level”. Det kan vara både nivå och ambition av test

planerad och man skall veta vilka delar man skall testa, exempelvis användningsfall och det skall finnas någon form av checklista för vad man är intresserad av.

Testplanen är ett synnerligen levande dokument och behöver ses över ofta, börja med en gång i veckan.

IEEE829:2008 innehåller en hel del praktiskt användbar information om vad man kan ha med i både testplan och testrapport. Standarden är mycket generisk och använder sig av ett nivå-begrepp som är tillämpbart för mycket stora organisationer, men om man skippar information om Master level, är informationen i kapitlen 9-13 bra checklistor.

https://doi.org/10.1109/IEEESTD.2008.4578383

ISO/IEC/IEEE 29119-3:2013 är en nyare standard. Den är mindre heltäckande, men å andra sidan innehåller den tips på olika dokument, deras disposition och vad man kan skriva om under varje rubrik. 6.2 innehåller testplan, 7.2 test design specifikation, 7.4 test procedur specifikation, 7.3 testfallsspecifikation

https://doi.org/10.1109/IEEESTD.2013.6588540

Testrapport

Testrapporten är en parhäst till testplanen och innehåller:

Testfall som körts och deras utfall. Testlogg.

Även här får man länka till resultat från olika verktyg för automatiserade testkörningar.

Data från metriker som insamlats.

Avvikelser och hur de hanteras.

Åtgärder. Kan vara länkar till Issues i Git.

Sammanfattad bedömning av produktens status.

Även här innehåller IEEE829:2008 och ISO/IEC/IEEE 29119-3:2013 värdefulla checklistor som man plockar tillämpliga delar ifrån. I ISO/IEC/IEEE 29119-3:2013 innehåller 6.3 Test status report, 7.9 Actual results, 7.10 Test result, 7.11 Test execution log och 7.12 test incident report

Feed-back om dokument

Man kan få tre typer av omdömen av dokument:

1. Bra, det kan finnas några tips på hur det kan bli bättre, men ni uppfyller kurskraven.
2. Komplettera till nästa gång. Dokumentet uppfyller inte förväntningarna till inlämningen, komplettera till nästa inlämningstillfälle.
3. Komplettera nu. Dokumentet innehåller stora fel, som kommer att ställa till det för er i projektet. Lämnas snarast till handledaren.