Handelshögskolan
VID GÖTEBORGS UNIVERSITET
Institutionen för informatik
2004-01-16

Användningen och nyttan av systemdokumentation

Abstrakt

Många är överens om vikten av att dokumentera vid systemutveckling. Systemdokumentation av god kvalitet är till stor hjälp när ett system skall underhållas och vidareutvecklas, dessutom är det ett utmärkt sätt att förbättra kommunikationen under systemutvecklingsarbetets gång. Men när utvecklingen är klar, dokumentationen skriven och projektet avslutat, vad händer då? Används systemdokumentationen? Det verkar finnas en utbredd allmän uppfattning dels att systemutvecklare inte tycker om att skriva dokumentation men också att det bland systemutvecklare finns en inbyggd aversion mot att läsa densamma. Syftet med denna studie var att belysa användandet av systemdokumentation med utgångspunkt från systemutvecklarna. Studien byggde på individuella djupintervjuer med systemutvecklare på fyra olika IT-företag. Studien visade att systemdokumentationen används, dock inte i någon större omfattning, istället används andra informationskällor, t.ex. muntlig information eller källkod. Detta beror på att tillgängligheten till systemdokumentationen ofta är dålig och att nyttan av systemdokumentationen upplevs som liten. Studien visade bl.a. på att de främsta orsakerna till detta var att systemdokumentationen till form och innehåll inte är anpassad för den tilltänkta användaren (systemförvaltaren) och att tillgängligheten till systemdokumentationen försämras av att man inte skiljer på projektdokumentation och produktdokumentation (systemdokumentation), varken rent fysiskt eller konceptuellt.

Nyckelord: systemdokumentation, dokumentation, inline-dokumentation, systemutveckling, produktdokumentation, projektdokumentation, dokumentationsstandard.

Författare: Åse Magnusson
Handledare: Helena Holmström
Examensarbete I, 10 poäng

Tack

Ett stort tack till alla systemutvecklare som ställde upp och lät sig intervjuas. Många lever under stor tidspress på arbetet och jag är tacksam över att de tog sig tid till att tala med mig. Tack också till Jonas som korrekturläst, kommit med nyttig feedback och ifrågasatt varje ord så att uppsatsen kunde utvecklas och förädlas. Sist men inte minst tack till min handledare Helena Holmström vars kommentarer bidrog till värdefulla insikter om komplexiteten i problemområdet.

1 Inledning

Många är överens om vikten av att dokumentera vid systemutveckling (Booch, 1994; Mathiassen et. al., 1998). Systemdokumentation av god kvalitet är stor hjälp när ett system skall underhållas och vidareutvecklas (Avison & Fitzgerald, 1995; Sommerville, 1995). Dessutom är det ett utmärkt sätt att förbättra kommunikationen under systemutvecklingsarbetets gång. Men när utvecklingen är klar, systemdokumentationen skriven och projektet avslutat, vad händer då? Används systemdokumentationen?

Det verkar finnas en utbredd allmän uppfattning om att systemutvecklare inte tycker om att skriva dokumentation (Mathiassen et. al, 1998; Wyssocky, 2003), och att det bland systemutvecklare finns en inbyggd aversion mot att läsa densamma. En av de främsta förespråkarna inom Agile-rörelsen[1], Scott Ambler (2001a), utgår från att dokumentation som produceras för systemutvecklarnas ögon, med undantag för inline-dokumentation, aldrig kommer att läsas. Han hävdar att: *"these people [developers] are not going to trust, let alone read, documentation outside of the code"* (Ambler, 2001a).

Vidare, i en magisteruppsats skriven på institutionen för informatik vid Göteborgs Universitet, redovisar Alexandersson och Johansson (2002) att mycket av systemutvecklarnas motvilja till att dokumentera grundar sig i en känsla att dokumentationen ändå aldrig blir läst. Många systemutvecklare tror att *"dokumentet sätts in i en pärm och kommer aldrig att läsas av någon"* (Alexandersson & Johansson, 2002, s.34).

Jag har även personligen, i mitt arbete som systemutvecklare, upptäckt att det finns en obalans mellan hur viktigt det anses att systemdokumentationen produceras (något som ofta betonas i mycket angelägna termer) och hur systemdokumentationen faktiskt används. Stämmer detta? Har de tillfrågade systemutvecklarna i Alexanderssons och Johanssons studie rätt i sina befarelser? Använder inte systemutvecklarna den systemdokumentation som finns? Om inte, vad beror det på?

1.1 Syfte

Syftet med denna studie var att belysa användandet av systemdokumentation med utgångspunkt från systemutvecklarna. Vetskapen om hur systemdokumentationen används eller varför den, i vissa fall, inte används kan klargöra bristerna i den systemdokumentation som produceras i dag och även eventuella brister i hanteringen av systemdokumentationen. Detta i sin tur kan leda till att den systemdokumentation som produceras blir av bättre kvalitet och mer tillgänglig för de tänkta läsarna och således används för det som det var tänkt; som en värdefull informationskälla vid förvaltning och vidareutveckling av system.

[1] Agile betyder ”rörlighet” och är ett samlingsnamn för lättviktsmetoder som betonar vikten av flexibilitet och fungerande mjukvara över hårt styrda processer och omfattande dokumentation (Ambler, 2001a; Highsmith, 2001).

1.2 Avgränsning och huvudfrågor

Jag valde att begränsa mig till den dokumentation som främst är till för systemutvecklare, vad jag definierar som systemdokumentation, och studien kommer inte att behandla användardokumentation och projektdokumentation. För mer information se kapitel 2 (Definitioner), där de olika dokumentationsbegreppen definieras mer utförligt. Studien fokuserade främst på systemutvecklarnas uppfattningar om existerande systemdokumentation och deras erfarenheter och upplevelser kring att använda densamma.
Frågorna jag utgick från inför min studie är:

På vilket sätt använder systemutvecklare systemdokumentation som informationskälla vid vidareutveckling och förvaltning?
Hur uppfattar systemutvecklarna nyttan av systemdokumentationen?

1.3 Disposition

I kapitel 2 (Definitioner) ges definitioner av dokumentation och systemdokumentation som begreppen används i denna studie. Vidare i kapitel 3 (Teoretisk referensram) ges en teoretisk referensram till studien. Dokumentationens roll i systemutvecklingen diskuteras och relevanta inriktningar beskrivs. Dessutom diskuteras kvalitet hos dokumentation, dokumentationsstandard och dokumentation i källkoden. I kapitlet redovisas även tidigare forskning gällande användningen av systemdokumentation. Kapitel 4 (Metod) redogör för metoden som används vid insamling av empiriskt data, urvalet av intervjuobjekt och hur intervjuerna gått tillväga, samt hur det empiriska materialet har analyserats. Dessutom så presenteras intervjuobjekten och deras företag med en kortfattad beskrivning. Resultatet av den empiriska studien presenteras i kapitel 5 (Resultat). I kapitel 6 (Diskussion) förs en diskussion om resultatet av den empiriska studien. Och slutligen, kapitel 7 (Slutsats) redogör för slutsats och utvärdering av studien samt ger uppslag till eventuella fortsatta studier. Referenser som finns i uppsatsen; böcker, artiklar, uppsatser och webbsidor, listas i kapitel 8 (Referenser).

2 Definitioner

I detta kapitel ges definitioner av dokumentation och systemdokumentation som begreppen används i denna studie.

2.1 Dokumentation

Dokumentation är ett vagt och brett begrepp. Att ge sig på en fullständig och entydig definition av begreppet är näst intill omöjligt eftersom innebörden av dokumentation kan variera kraftigt beroende på typ av IT-system och mellan olika organisationer. Dock känner jag ett behov av att redogöra för min definition av systemdokumentation; den definition som jag hade som utgångspunkt i min studie. Jag har utgått från Sommervilles (1992) definition av dokumentation, eftersom den i detta avseende stämmer väl in på min egen uppfattning, men jag har även fördjupat definitionen av systemdokumentation något.

Generellt kan dokumentation delas in i två kategorier: dokumentation som relaterar till projektet och processen, ***projektdokumentation***, och dokumentation som relaterar till produkten (d.v.s. systemet), ***produktdokumentation***.
Projektdokumentation kan t.ex. bestå i projektplan, projektbeskrivning och dylikt, och denna grupp av dokumentation kommer inte att behandlas i denna studie. Produktdokumentationen skall beskriva systemet och bör inte ses som en oberoende enhet, utan som en del av systemet som den har till uppgift att beskriva. Detta stämmer väl in på Forwards definition som lyder:

> *[Software documentation is] an artifact whose purpose is to communicate information about the software system to which it belongs.* (Forward, 2002, s.1)

Vidare kan produktdokumentationen indelas i två underkategorier: ***användardokumentation*** och ***systemdokumentation***. Användardokumentation är till för användarna av systemet, och beskriver således hur systemet skall användas. Systemdokumentation beskriver hur systemet är uppbyggt, och riktar sig främst till systemutvecklare. I denna uppsats behandlar jag endast systemdokumentation. Användardokumentation och rent projektrelaterad dokumentation diskuteras således inte, inte heller dokumentation över system eller komponenter från tredje part. Och därför definieras endast begreppet systemdokumentation i närmare detalj nedan.

2.1.1 Systemdokumentation

Systemdokumentation har således som sin uppgift att beskriva hur systemet är uppbyggt och kan i sin tur delas upp i två varianter: ***extern systemdokumentation*** och ***inline-dokumentation*** [2]. Med extern systemdokumentation menas all systemdokumentation som existerar utanför systemet, i form av t.ex. textdokument eller diagram. Inline-dokumentation är den systemdokumentation som går att finna inuti systemet i form av kommentarer i källkoden. Syftet med inline-dokumentation kan t.ex. vara att dokumentera en klass API eller att förklara och belysa vissa implementationsval för att öka förståelsen för koden vid en genomläsning. En annan viktig aspekt är att informera om att en förändring i ett visst källkodsavsnitt kan få konsekvenser i andra delar av systemet. Inline-dokumentation kan bestå av enkla

[2] Jag har valt att använda det engelska begreppen inline eftersom det är det begrepp som används mest frekvent bland systemutvecklare idag.

förklarande kommentarer i koden eller mer standardiserade kommentarer, t.ex. i xml-format, med syfte att automatgenerera hjälpfiler eller dylikt. Inline-dokumentation kan således leda till skapandet av extern systemdokumentation. Oavsett hur dokumentationen tas fram så kan man enkelt särskilja dessa två varianter av systemdokumentation.

3 Teoretisk referensram

I detta kapitel ges en teoretisk referensram till studien. Dokumentationens roll i systemutvecklingen diskuteras och relevanta inriktningar beskrivs. Dessutom diskuteras kvalitet hos dokumentation, dokumentationsstandard och dokumentation i källkoden. Slutligen redogörs för tidigare forskning gällande användningen av systemdokumentation.

3.1 Systemdokumentationens roll i systemutvecklingsprocessen

Lika länge som systemutveckling funnits som en disciplin har systemdokumentation varit livligt debatterat och således har det skrivits en hel del om detta ämne. De flesta är nog överens om att dokumentationen är en viktig del av utvecklingsarbetet, men åsikterna går isär kring vad och hur mycket som skall dokumenteras, och hur denna dokumentation skall produceras. En stor del av kostnaderna för utvecklingsprojekt består i produktion av allehanda dokumentation. Felaktigheter eller avsaknad av information i denna dokumentation kan leda till att systemet används och driftas fel med kostsamma driftsstopp som följd. Därför anser Sommerville (1992) att det är viktigt att ägna lika mycket uppmärksamhet åt dokumentationen som det resulterade systemet. Ett system består inte enbart av de körbara programmen utan inbegriper även all dokumentation som behövs för att använda och förvalta systemet (Sommerville, 1992). Därför kan inte systemet anses som klart innan all dokumentation är färdigställd (Nickerson, 2001).

Vad som skall dokumenteras om ett system beror på många olika faktorer. Ofta finns det krav på vad som skall dokumenteras från kund, myndigheter eller den egna organisationen. Sedan beror det naturligtvis på hur verksamhetskritiskt systemet är och komplexiteten i lösningen. Man få även ta i beaktande hur länge systemet beräknas vara i drift – ett system som förväntas ha en kort livslängd är det kanske inte berättigat att dokumentera ingående (Booch refererad i Brown, 1997; McConnell, 1996). När ett system förväntas ha en lång livscykel däremot, och förvaltning och vidareutveckling förväntas överstiga kostnaden för utvecklingen (Jia, 2000), är det klokt att lägga ner mer arbete på att producera dokumentation.

Allmänt kan man säga att den externa systemdokumentationen bör prioritera översikt framför detaljer (Booch, 1994); övergripande diagram över systemets arkitektur och argument för valda designmönster framför detaljerade beskrivningar om funktioner och subrutiner. McConnell (1996) anser att grafiska illustrationer är lättare att förstå, speciellt vid kommunikation med användare och kund, och att man därför bör fokusera dokumentationen kring diagram. Många anser att en komplett kravspecifikation är en av de viktigaste delarna i ett systems dokumentation (Sommerville, 1995). I kravspecifikationen dokumenteras alla funktionella och icke-funktionella krav på systemet. Annan viktig information, som bör vara med i den externa systemdokumentationen såväl som inline-dokumentationen, är information om brister och problem i systemet (McConnell, 1996; Sommerville, 1992).

> *"Don't make people discover defects on their own when you know about them already: document known limitations."* (McConnell, 1996, s.534).

Någonting som ytterligare komplicerar situationen är att dokumentationen har flera olika syften under ett systems livstid och följaktligen flera tilltänkta läsare.

Under utvecklingsprojektet så skall dokumentationen stödja kommunikation, såväl mellan projektmedlemmar som mellan projektet och kunden, och fånga krav och beslut gällande systemet. Dokumentationen kan även fungera som styrredskap för projektets framåtskridande och agera som informationsbas för projektledare och chefer vid planering och budgetering (Sommerville, 1992). Dessutom är dokumentationen viktig för att synliggöra utvecklingsarbetet. Ett system är en abstrakt entitet och under stora delar av utvecklingsprojektet finns det inget synligt system att relatera till och därför spelar dokumentationen en roll i att synliggöra och konkretisera arbetet (Sommerville, 1995). Dokumentationen skall också beskriva för slutanvändarna hur systemet skall användas och administreras (Sommerville, 1992). När systemet sedan är i drift skall det finnas dokumentation över dess implementation för att möjliggöra förvaltning och eventuell vidareutveckling (Sommerville, 1992).

Dokumentationens roll i utvecklingsprocessen skiljer sig också mellan olika utvecklingsmetoder. Vattenfallsmodellen (som länge var dominerande) består i ett antal faser som följer på varandra utan att överlappa, d.v.s. att en fas måste vara helt avslutad innan nästa påbörjas, och där varje fas resulterar i ett dokument som blir till input till nästa fas i processen. Dokumenten fungerar som kontrakt mellan två på varandra följande faser. Vattenfallsmodellen kan därför sägas vara en ”dokumentdriven” modell (McConnell, 1996). Ett av de största problemen med vattenfallsmodellen är att den inte är tillräckligt flexibel; att det är kostsamt att gå tillbaka och göra ändringar när förutsättningar förändrats (Avison & Fitzgerald, 1995; McConnell, 1996).

De flesta organisationer har numera gått ifrån vattenfallsmodellen i och med att medvetenheten har ökat angående utvecklingsprocessens iterativa natur och vikten av att hantera förändringar i kraven under utvecklingens gång (Avison & Fitzgerald, 1995; McConnell, 1996). De senaste åren har utvecklingsmetoden RUP[3] och modelleringsspråket UML[4] fått stor genomslagskraft och används i stor omfattning runt om i IT-världen. RUP är en välstrukturerad metod som bygger på en iterativ utvecklingsprocess. RUP anses vara mer flexibel än metoder som bygger på vattenfallsmodellen eftersom den bygger på en iterativ utvecklingsprocess och inkrementell integration och det är tillåtet för faserna att överlappa varandra (Krutchen, 2000/2002). Metoden innehåller riktlinjer gällande roller (vem?), aktiviteter (hur?), arbetsflöden (när?) samt leverabler (vad?) för alla faser under utvecklingsarbetet. Även i RUP spelar dokumentationsarbetet en central roll och bidrar till att utvecklingsprocessen drivs framåt. RUP innehåller klara riktlinjer kring dokumentation: vilka dokument och modeller som skall produceras, hur och när de skall produceras och av vem. Dessutom så innehåller metoden mallar och checklistor för den föreskrivna dokumentationen.

På senare tid har det vuxit fram en rörelse som kan ses som en reaktion mot RUP och andra starkt strukturerade utvecklingsmetoder (som ofta innebär en omfattande produktion av dokumentation), nämligen Agile-rörelsen. Agile är ingen utvecklingsmetod utan ett samlingsnamn för ett antal lättviktsmetoder (t.ex. Extreme

[3] Rational Unified Process är en utvecklingsmetod som utvecklats och underhålls av Rational Software (Krutchen, 2000/2002).
[4] Unified Modeling Language är ett visuellt modelleringsspråk som används för att representera olika slags IT-system (Eriksson & Penker, 1998).

Programming[5]). Agile har som utgångspunkt att systemet bör sättas i första rummet och att det först och främst är kunden som avgör vilken dokumentation man är beredd att betala för. Agile betonar devisen *”travel light”* som innebär att inte producera för stora mängder dokumentation tidigt i utvecklingsprocessen för att på så sätt underlätta snabba ändringar och bibehålla hög flexibilitet i projektet, att vara ”Agile”. *”You can't expect to carry a lot of baggage and move fast.”* (Beck, 2000, s. 42).
Detta förhållande till dokumentation missförstås ofta och Agile-utvecklare avfärdas som ”hackers” som gör allt för att slippa dokumentera. Agiles förespråkare menar dock inte att man skall strunta i att dokumentera, men att man skall låta bli att dokumentera när det inte absolut behövs (Ambler, 2001a), och att omfattningen av dokumentationen bör begränsas. *“We embrace documentation, but not hundreds of pages of never-maintained and rarely-used tomes.”* (Highsmith, 2001).
Agile-förespråkarna lever mycket enligt mottot *”just good enough”* (Ambler, 2001a) när det gäller dokumentation. Dock understryks modellering som en viktig del i utvecklingsarbetet. Agile Modeling (AM) är en praktisk metod för modellering som kan användas i kombination med, eller som komplement till, andra utvecklingsmetoder, t.ex. RUP (Ambler, 2001b).

Även Booch (1994) tillstår att dokumentation visserligen är viktigt, men ändå underordnat utvecklandet av själva systemet och produktionen av dokumentation bör inte driva utvecklingsprocessen. Booch (refererad i Brown, 1997) anser att en organisation inte bör spendera mer än 5-10 % av den totala utvecklingstiden på att producera dokumentation, och att endast system som är av mycket kritisk natur; t.ex. där människors liv är beroende av systemet, behöver dokumenteras ingående. Oavsett vilken ståndpunkt man har gentemot dokumentation så är och förblir dokumentationen en viktig del av systemutvecklingsprocessen och framför allt nödvändig för att kunna vidareutveckla och förvalta ett system.

3.2 Kvalitet hos dokumentation

Vad är god dokumentation? Om bristande kvalitet hos dokumentationen leder till att den inte används så är tiden och kostnaden för att ta producera den bortkastad. Sommerville (1992) anser att kvaliteten på dokumentationen är lika viktig som kvaliteten på systemet. Vidare förklarar han att det varken är enkelt eller billigt att producera dokumentation av god kvalitet, eftersom det kräver engagemang både av utvecklare och av organisationen. Det krävs kompetens och disciplin av systemutvecklarna och det krävs att man har en dokumentationsstandard och en kvalitetssäkringsprocess i organisationen som fungerar och efterlevs. Sommerville (1992) anser också att dokumentationen bör granskas. Systemet genomgår ju ett antal noggranna tester innan det anses vara godkänt och kan levereras till kund, och på samma sätt bör dokumentationen granskas.

Mathiassen et.al. (1998) anser att det som utmärker bra dokumentation är ***klarhet*** och ***elegans***. Med klarhet avses hur lättläst och lättförståelig dokumentationen är. Saker som kan bidra till klarhet är enligt Mathiassen et. al.: ett standardiserat utseende, konsekvent användande av tekniska begrepp, ett komplett index och eventuellt en ordlista. Standardiserade dokument och modeller är ofta lättare att förstå och läsa

[5] Extreme Programming är en utvecklingsmetod som går under samlingsnamnet Agile. Denna metod beskrivs ofta som en ”testdriven” utvecklingsmetod (Beck, 2000).

(Sommerville, 1992; Mathiassen et.al., 1998) och även ett dåligt strukturerat dokument kan bli användbart om det har ett komplett index (Sommerville, 1992). Med elegans åsyftar Mathiassen et. al. (1998) relevans; att dokumentationen innehåller relevant information för läsaren. Vikten av att skriva kortfattad dokumentation understryks också av Mathiassen et. al. (1998), som menar att man endast då kan uppnå hög kvalitet på dokumentationen.

Många har känt sig manande att bidra med metoder, verktyg och koncept för framtagandet av god dokumentation. Ett exempel som inte har fått någon större genomslagskraft i IT-världen är Literate Programming (Smith, 2001). Literate Programming är en teknik för att dokumentera IT-system som togs fram av Donald Knuth i början på 1980-talet. Idén med Literate Programming är att programmeraren inte skall fokusera på att skriva instruktioner till datorn, utan på att få människor att förstå vad man instruerar datorn att göra. I Literate Programming förvaras dokumentation och källkod i samma fil i en form som lämpar sig för mänsklig förståelse. Literate Programming-verktyg används för att tolka filen och producera dokumentation respektive kompilerbar källkod.

På marknaden finns det ett antal verktyg som säger sig skapa en komplett systemdokumentation från enbart okommenterad källkod. Dokumentationen blir inte i form av diagram utan i Word-, pdf- eller html-format. Exempel på sådana verktyg är Visual Expert (Novalys) och Doc-O-Matic (toolsfactory, 2003).
Problemet med dessa verktyg är att dokumentationen som skapas enbart innehåller fakta som kan tas direkt från källkoden, d.v.s. beskrivningar om klasser och komponenter med ingående funktioner, den systemdokumentation som anses minst viktig och vissa fall helt redundant i externa dokument (Booch, 1994). Dessa slags ”dokumentationsgeneratorer” kan därför endast ses som ett komplement till övrig systemdokumentation och inte som en lösning på dokumentationsproblemet.

Det har också påbörjats ett arbete med att ta fram en modell för utvärdering av mogenheten i dokumentationsprocessen, Documentation Maturity Model (DMM) (Huang & Tilley, 2003), motsvarande en modell som finns för programutveckling, Capability Maturity Model[6] (CMM). En av tankarna bakom DMM (och CMM) är att när en organisation förbättrar sin utvecklingsprocess så förbättras även kvaliteten på den resulterande produkten (dokumentation respektive mjukvara). I DMM skiljer man mellan två olika aspekter av dokumentationskvalitet: kvalitet gällande dokumentationsprocessen och kvalitet gällande själva dokumentationen; dokumenten eller modellerna. Arbetet med DMM är dock ännu i sin linda och det är oklart vad arbetet skall resultera i.

Ett stort problem är att underhålla dokumentationen; att hålla dokumentationen uppdaterad så att den ger en korrekt spegling av systemets implementation. Alltför ofta är dokumentationen inaktuell redan när systemet tas i drift och därefter uppdateras den sällan (Forward, 2002; Sommerville, 1992). I detta hänseende har mycket tilltro satts till diverse CASE-verktyg. CASE-verktyg kan bland annat användas för att generera källkod utifrån modeller av systemet, vad som populärt kallas ”forward engineering” och det motsatta ”reverse engineering”, att man från källkod kan skapa t.ex. klassdiagram eller sekvensdiagram. De senare åren har det

[6] CMM utvecklades av Software Engineering Institute vid Carnegie Mellon universitetet. Kallas även för Capability Maturity Model for Software, SW-CMM (SEI, 2003).

även kommit fram fungerande s.k. ”roundtrip engineering-verktyg” som, till skillnad från reverse- respektive forward engineering-verktyg, verkar åt båda håll. Källkod kan genereras utifrån modeller av systemet och när källkoden sedan förändras uppdateras även modellerna. Exempel på ”roundtrip engineering-verktyg” är Rational XDE (Rational) och Borland Together (Borland, 2003).

Även om användning av CASE-verktyg kan bidra till att effektivisera dokumentationsarbetet så kan det inte ses som en allmän kvalitetshöjare på varken systemet eller dokumentationen. *”CASE can help…[b]ut it isn’t a silver bullet.* (McConnell, 1996, s.366)”. Dessutom är CASE-verktyg ofta väldigt kostsamma, både till priset men också när man beaktar utbildning och support för verktyget (Avison & Fitzgerald, 1995).

Forward (2002) anser att det i debatten om dokumentationens kvalitet fokuseras för mycket på problemet med att dokumentationen inte är uppdaterad och komplett. Han hävdar att det finns andra egenskaper som mer bidrar till hög kvalitet hos dokumentationen.

> *“Software documentation should focus more on conveying meaningful and useful knowledge than on precise and accurate information.”* (Forward, 2002, s.79).

Forwards studie visar också på att dokumentationen inte nödvändigtvis måste vara helt uppdaterad för att fylla ett syfte och upplevas som användbar.

3.3 Dokumentationsstandard

Som nämnts i föregående avsnitt anser Sommerville (1995) att tillämpning av dokumentationsstandarder är avgörande för att producera dokumentation av god kvalitet. Sommerville (1995) delar upp dokumentationsstandard i tre olika typer: processtandard, produktstandard och ”interchange”-standard. Med ***processtandard*** menar han den standard som definierar hur och när dokumentationen skall produceras, t.ex. vilka dokument som är aktuella vid en viss fas i utvecklingen och hur man skall gå till väga för att skapa dokumenten. ***Produktstandard*** definierar dokumenten och modellernas utformning, t.ex. namngivningsregler, standardiserade dokumentmallar och vilken notation som skall användas vid modellering. ***Interchange-standard*** handlar om i vilket format dokument och modeller skall lagras. Detta är viktigt för att dokumentationen skall kunna användas av samtliga intressenter i ett utvecklingsprojekt och under efterföljande drift- och förvaltningsfas.

Utvecklingsmetoden RUP som nämnts ovan innehåller en dokumentationsstandard som består av både en processtandard och en produktstandard enligt Sommervilles indelning (Krutchen, 2000/2002). Det finns även dokumentationsstandarder som tagits fram av internationella organisationer såsom IEEE (Institute of Electrical and Electronics Engineers) och ISO (International Organization for Standardization), t.ex. ISO/IEC 6592. För mer information hänvisas till IEEE (IEEE, 2003) och ISO (ISO, 2003) på Internet. Det är dock inte nödvändigt att använda en officiell standard. Många organisationer har idag definierat och implementerat egna dokumentationsstandarder som är anpassade efter deras behov.

3.4 Dokumentation i källkoden

Inline-dokumentation har också varit ett livligt debatterat ämne. Wyssocky (2003) menar att det bland debattörerna finns det en tydlig uppdelning i två läger: de som enbart ser kommentarer i källkoden som störande och redundant information och de som anser att inline-dokumentationen är en värdefull informationskälla som bidrar till ökad förståelse av systemet. Förespråkarna för den första ståndpunkten hävdar att välskriven källkod är självförklarande och att alla kommentarer därför är redundanta och tar fokus från den verkliga informationen, källkoden. De som förespråkar inline-dokumentation anser att även fast källkoden är välskriven och till viss del självförklarande, framgår inte all information av källkoden. Till exempel framgår inte varför man valt en viss implementation av källkoden; ur källkoden (hur självförklarande den än är) kan man endast utläsa hur man gjort en viss implementation, inte varför. Ett annat argument mot inline-dokumentation som man ofta hör i debatten är att det är svårt att hålla källkoden och kommentarerna synkroniserade och att det är bättre att inte ha några kommentarer alls än kommentarer som inte stämmer överens med den faktiska implementationen. Wyssocky tillstår att det ligger en del sanning i argumentet men att detta problem är vanligast med kommentarer som beskriver hur någonting implementerats. Inline-kommentarer som redogör för varför en viss implementation är vald är det inte lika vanligt att man måste ändra. Och som Wyssocky redan har konstaterat så är kommentarer som svarar på frågan varför? att föredra framför kommentarer som svarar på frågan hur?, eftersom hur man implementerar en lösning redan framgår av koden. McConnell (1993) menar att eftersom inline-kommentarer är så tätt knutna till källkoden så är det större sannolikhet att inline-dokumentationen överensstämmer med verkligheten än annan dokumentation. Det finns dock en risk att inline-dokumentation missbrukas; att det kommenteras alltför ymnigt och självklara lösningar kommenteras. Det enda syftet med inline-dokumentation bör vara hjälpa utvecklare att förstå källkoden vid förvaltning och vidareutveckling av systemet (Wyssocky, 2003), och därför är det viktigt att rätt saker kommenteras på ett snyggt och lättförståeligt sätt så att inte kommentarerna motverkar sitt syfte och gör mer skada än nytta; *”[p]oor comments are worse than no comments”* (McConnell, 1993, s.463). McConnell (1993) anser också att det är viktigt att inline-dokumentationen har en konsekvent form som är effektiv och enkel att uppdatera, och att man därför bör undvika avancerade formationer med stjärnor eller dylikt. Dessutom bör källkoden kommenteras kontinuerligt under programmeringen. På så sätt så blir kommentarerna av bättre kvalitet och kommenteringen tar mindre tid i anspråk än om man skriver alla kommentarer när programmet redan är klart (McConnell, 1993).

3.5 Användningen av systemdokumentation

Om och hur dokumentationen sedan används har inte fått lika mycket uppmärksamhet i debatten. Enstaka undersökningar har dock gjorts som redogör för användningen av systemdokumentation, t.ex. Forward (2002), vars studie främst belyser vilken typ av dokumentation som används, hur ofta den används och vem som använder den (systemutvecklare eller projektledare). Forwards studie redogör även för preferenser för olika dokumentationsverktyg och behovet av verktyg som åskådliggör diskrepans mellan ett systems implementation och dess dokumentation. Forward presenterar även en modell för att förutsäga en dokumentationsartefakts (t.ex. dokument eller modell) relevans med hänsyn till ”usefullness”, ”referential decay” och ”authority”. Med ”usefullness” avser Forward (2002) användbarhet; hur

väl ett dokument kan tillgodose användarens behov av kunskap och information i relation till andra dokument. Forward tillstår att detta inte är ett helt objektivt mått eftersom det baseras på hur ofta ett dokument används och av vem samt på individuell feedback från användarna. ”Referential decay” är ett mått som belyser sannolikheten att ett dokument innehåller inkonsekvent information i relation till andra dokument. Slutligen, ”authority” är ett mått på i vilken omfattning ett dokument innehåller auktoritativ information och vilken omfattning ett dokument refererar till auktoritativ information i andra dokument.

Ett annat exempel, som tangerar användningen av systemdokumentation, är ett arbete (som påbörjats men som inte är klart) med att förbättra och effektivisera förvaltningsarbetet genom att undersöka vilka informationskällor som används vid förvaltning (Seaman, 2003). Som del i detta arbete har Seaman genomfört en studie i hur systemförvaltare skaffar sig information om systemet de satts att förvalta och vilka informationskällor de anser vara mest användbara (2002).

Forward (2002) efterfrågar mer forskning av kvalitativ natur kring hur systemdokumentationen används i praktiken. Jag skall med denna studie ge mitt bidrag.

4 Metod

I detta kapitel redovisas metoden som använts vid insamling av empiriskt data, urvalet av intervjuobjekt och hur intervjuerna gått tillväga, samt hur det empiriska materialet har analyserats. Dessutom presenteras intervjuobjekten och deras företag med en kortfattad beskrivning.

4.1 Metodval

Vilken metod man skall använda vid insamling av material beror på flera faktorer. Hur ser frågeställningen ut? Vilka resurser finns? Forskarens erfarenhet? (Trauth, 2001). Syftet med studien måste främst styra valet av undersökningsmetod och aldrig tvärtom (Andersson, 1985). Dessutom måste man ta i beaktande olika metoders ***validitet*** och ***reliabilitet*** i hänsyn till det valda undersökningsområdet. Med validitet menas hur väl metoden undersöker det man ämnar undersöka och med reliabilitet menas hur väl metoden undersöker det man undersöker; med andra ord *”metodens känslighet för slumpinflytelser”* (Andersson, 1985, s.22). Det vanligaste sättet att kategorisera forskningsmetodik är indelningen i ***kvantitativa*** respektive ***kvalitativa*** undersökningsmetoder (Backman, 1998). Kvantitativa metoder syftar till att få in data som kan omvandlas till siffror och grafer som man sedan kan genomföra statistiska analyser på. Den kvalitativa undersökningsmetoden består i att samla in rik och ostrukturerad data (av ickekvantifierbar natur) som ofta går på djupet av ett problemområde och sedan tolka denna utifrån egna erfarenheter och uppfattningar (Backman, 1998). Innehållet i kvalitativ data blir således mycket mer svårfångat och analysen och tolkningen kräver mer dedikation och erfarenhet av utövaren (Backman, 1998). Dessutom så krävs det viss branschkännedom (i detta fall IT-branschen) och praktisk insikt i problemområdet (i detta fall systemdokumentation) för att kunna tolka och förstå det kvalitativa materialet rätt.

Inriktningen på en studie kan vara antingen ***induktiv*** eller ***deduktiv***. Med induktion menas att teorier eller hypoteser skapas utifrån det insamlade materialet, efter en noggrann analys. Deduktion däremot innebär att forskaren skapat sig en teori eller hypotes innan insamlingen av data påbörjas och att studien syftar till att antingen styrka eller falsifiera denna hypotes eller teori (Backman, 1998). En induktiv studie kan därför sägas vara ”*hypotesgenererande”* och en deduktiv studie kan sägas vara ”*hypotesprövande”* (Backman, 1998, s.48).

Jag valde en kvalitativ undersökningsmetod eftersom min studie främst grundas på de intervjuade systemutvecklarnas personliga uppfattningar och erfarenheter, begrepp som är svåra att kvantifiera (Strauss & Corbin, 1998). Jag kommer således att presentera resultatet av undersökningen i löpande text. Undersökningen hade en induktiv inriktning; jag utgick inte från någon teori eller hypotes utan lät det empiriska materialet styra formandet av eventuella teorier. Syftet med studien var främst deskriptivt, d.v.s. jag ämnade först och främst beskriva och redogöra för systemutvecklarnas uppfattningar och erfarenheter kring att använda systemdokumentation. Jag hade huvudsakligen ingen normgivande intention, men jag kan dock, efter analys av materialet, komma till slutsatser som leder till att jag bidrar med förslag till förbättringar för att öka nyttan och användningen av systemdokumentation.

Jag valde individuella djupintervjuer eftersom jag ansåg det vikigt att informationen blev så komplett och relevant som möjligt, och eftersom intervjuer är ett tillvägagångssätt som ger stort utrymme för följdfrågor och förtydliganden (Andersson, 1985).

4.2 Urval

Min studie byggde på djupintervjuer av systemutvecklare. Systemutvecklarna har haft varierande arbetsuppgifter så som: nyutveckling, support, vidareutveckling och förvaltning av system. Urvalet grundade sig på att få systemutvecklare från flera olika slags IT-företag och om möjligt med varierande bakgrund. Mitt mål med urvalet var även att få en jämn könsfördelning. Detta har jag dock inte lyckats med; endast en av de intervjuade systemutvecklarna är kvinna. Totalt har jag intervjuat sex systemutvecklare på fyra olika företag. Intervjupersonerna är anonyma och deras namn återfinns inte i mitt material. Även företagen är anonyma och nedan återfinns en kortfattad beskrivning av de olika företagen och intervjupersonerna.

4.2.1 Beskrivning av företag i studien

Företag A: IT-avdelning som ingår i en större koncern i logistikbranschen. IT-avdelningen har ca. 15 anställda och har funnits sedan 1999. IT-avdelningen håller som bäst på att utveckla en egen utvecklingsmetod som skall innehålla policy för dokumentation. Notationen som används vid modellering är UML.

Företag B: Liten reklambyrå med begränsad systemutveckling. Företaget har totalt knappt 30 anställda varav 4 är systemutvecklare. Företaget har funnits i ca 15 år men har inte sysslat med systemutveckling hela den tiden. Systemutvecklingen består mest av webbapplikationer i Java och de arbetar enligt en egenutvecklad ”lättviktsmetod”. Notationen som används vid modellering är UML.

Företag C: Medelstor IT-konsultföretag som har funnits sedan 1995. Företaget har totalt ca. 400 anställa. Avdelningen i Göteborg har ca. 80 anställda varav ca. 25 är systemutvecklare. Arbetar nästan uteslutande med webbprojekt ”in-house”. Företaget har en egen utvecklingsmodell som är influerad av RUP. Modellen innehåller bl.a. mallar och checklistor för dokumentation. Notationen som används vid modellering är UML.

Företag D: Företaget har totalt ca 2000 anställda. Den aktuella avdelningen här i Göteborg har 25 anställda varav 20 är systemutvecklare. Företaget är inte ett renodlat IT-företag utan har flera affärsområden och har funnits i ca. 40 år. Företaget är certifierat enligt ISO 9001:2000[7] och TickIT[8]. Arbetar enligt projektmetoden PPS[9].

[7] ISO 9001:2000 är en internationell kvalitetssystemstandard (ISO, 2003).
[8] TickIT är en tolkning av ISO 9000 serien som skall hjälpa företag att bättre tillämpa detta kvalitetssystem (BSI, 1995; Paulk, 1994).
[9] PPS (Praktisk ProjektStyrning) är en styrmetod som utvecklas och underhålls av TietoEnator (TietoEnator, 2003).

4.2.2 Beskrivning av intervjupersoner i studien

Intervjuperson 1: Man som arbetar på företag A. Har arbetat som systemutvecklare i drygt 3 år varav ca 1 år på företag A. Arbetsuppgifterna består främst i nyutveckling av webbapplikationer i .Net. Viss förvaltning och vidareutveckling förekommer.

Intervjuperson 2: Man som arbetar på företag C. Har arbetat som systemutvecklare i 3 ½ år hela tiden på företag C. Arbetsuppgifterna består i support, förvaltning och vidareutveckling av existerande sajter och webbapplikationer.

Intervjuperson 3: Man som arbetar på företag D. Har arbetat som systemutvecklare i 18 år, hela tiden på företag D. Arbetsuppgifterna består främst i att förvalta och vidareutveckla ett specifikt system samt viss nyutveckling. Arbetet innebär traditionell mjukvaruutveckling.

Intervjuperson 4: Man som arbetar på företag B. Har arbetat som systemutvecklare i ca 3 år och har arbetat på företag B lika länge. Arbetsuppgifterna består i nyutveckling, vidareutveckling och visst underhåll av webbapplikationer i Java.

Intervjuperson 5: Kvinna som arbetar på företag C. Har arbetat som systemutvecklare i drygt 2 år, lika länge på företag C. Arbetsuppgifterna består främst i nyutveckling och viss vidareutveckling av webbapplikationer.

Intervjuperson 6: Man som arbetar på företag C. Har arbetat som systemutvecklare och systemarkitekt i 15 år, varav snart 4 år på företag C. För tillfället arbetar han främst med systemarkitektur och viss systemutveckling av webbaserade system. Tidigare har han arbetat med traditionell utveckling, bland annat på företag D.

4.3 Intervjufrågor

Vid formuleringen av mina intervjufrågor och även vid själva intervjutillfället hade jag gott stöd av boken *Som man frågar får man svar* (Andersson, 1985). I Anderssons bok finns bl.a. allmänna riktlinjer och checklistor att använda vid formulering av intervjufrågor, vad man bör tänka på och vilka fallgropar som finns. Innan intervjuerna hade jag tagit fram en frågemall med 26 frågor. Frågemallen var halvstrukturerad med öppna fristående frågor. Jag valde öppna frågor eftersom jag ville ge intervjupersonerna utrymme att svara så utförligt och nyanserat som möjligt på frågorna. Som stöd vid själva intervjutillfället hade jag även fördefinierat ett antal tänkbara svar till varje fråga. Jag var dock noga med att inte låsa mig vid dessa svar utan lät intervjupersonernas faktiska svar modifiera listan allt efter hand. Frågorna var initialt indelade i tre delområden:

- Systemutvecklarens bakgrund.
- Uppgifter om företaget.
- Upplevelser och erfarenheter kring att använda och ta till sig systemdokumentation.

Dock framgick det ganska snart att för att få fram all den information jag var intresserad av krävdes det specifika frågor angående inline-dokumentation. Därför tillkom en fjärde grupp frågor som behandlade inline-dokumentation separat.

4.4 Intervjuer

Jag valde att göra individuella djupintervjuer för att samla in data till studien. Intervjuerna ägde rum under en 4-veckorsperiod, inte mer än två intervjuer per vecka. I de fall där jag kände att jag efter intervjun ändå saknat viss information, som inte framgick av intervjusvaren, har jag fått kompletterande information från intervjuobjekten via e-post. Varje intervjutillfälle inleddes med att jag redogjorde för ämnet och syftet med undersökningen, samt gav min definition av systemdokumentation. Dessutom informerades alla intervjuobjekt om anonymiteten både för dem själva och för företaget, och de tillfrågades om det gick bra att intervjun bandades. Intervjuerna ägde oftast rum på den intervjuades arbetsplats i ett konferensrum eller liknande. Vid ett par tillfällen så genomfördes intervjuerna på ett kafé. Intervjuerna tog mellan 30 minuter och en timma att genomföra. Under intervjuns gång undvek jag i möjligaste mån att namnge olika dokument eller diagram direkt eftersom begrepp och namn ofta skiljer sig mellan olika standarder och företag. Vid några av intervjutillfällena var jag insatt i den gällande standarden och begreppen och använde mig av dem vid behov.

4.5 Analys av data

Alla intervjuerna bandades och så snart som möjligt efter slutförd intervju gjordes en ordagrann utskrift[10] av hela intervjun. Jag analyserade därefter intervjun och sökte efter gemensamma teman bland intervjusvaren. Intervjuerna analyserades löpande allt efter som de ägde rum. På så sätt upptäckte jag nya och intressanta perspektiv som kunde förbättra undersökningen och frågemallen kunde förädlas inför nästa intervju. Jag fann även att det var enklare och framför allt mer effektivt att göra analysen när intervjun fortfarande var i färskt minne. De teman jag fann genomgick en kontinuerlig uppdatering och förädling allt efter som intervjuerna analyserades.

Jag fann följande gemensamma teman i mitt empiriska material:
Användning av systemdokumentation
Nyttan av systemdokumentation
Tillgänglighet till systemdokumentation
Inline-dokumentation

I nästföljande kapitel kommer jag att redogöra för resultatet av den empiriska studien med utgångspunkt från ovanstående teman.

[10] Fullständiga textutskrifter av intervjuerna finns att tillgå mot förfrågan.

5 Resultat

I detta kapitel redovisas resultatet av den empiriska studien med utgångspunkt från följande teman: användning av systemdokumentation, nyttan av systemdokumentation, tillgänglighet till systemdokumentation och inline-dokumentation.

5.1 Användning av systemdokumentation

Majoriteten av de intervjuade systemutvecklarna använder och uppskattar ofta den externa systemdokumentationen. Inför uppgiften att sätta sig in i ett nytt system så letar de först och främst efter den systemdokumentation som finns. Det upplevs som viktigt att *”ha nånting att börja med, så man kan komma igång och felsöka”*. Den externa systemdokumentationen används oftast för att få överblick över systemet. Eller som en av de intervjuade uttrycker det:

> *Dokumentationen för mig är bara en överblick och en inkörsport - ett litet snabbt steg och en liten känsla [för hur det var tänkt].*

Systemutvecklarna vill främst ha övergripande konceptuella diagram som redogör för miljö i vilken systemet verkar, vilka komponenter som ingår och hur de relaterar till varandra. De gånger utvecklarna använder den externa systemdokumentationen för att inhämta information av mer detaljerad art handlar det främst om specifik driftsinformation.

> *Översikt tycker jag är absolut viktigast när det gäller dokumentationen och för att hitta saker…och detaljer är viktigt när man ska konfigurera sina miljöer och så…*

Den mer detaljerade kunskapen om systemets implementation föredrar många att hämta direkt från källkoden och dess inline-dokumentation.

> *…om man vill veta mer detaljer så tycker jag att man kan gå in och titta i källkoden lika gärna, det går ju fortare…*

Studien visar även på en skillnad mellan hur systemutvecklare med olika arbetsuppgifter använder systemdokumentationen. De utvecklare som har som sin främsta uppgift att förvalta och rätta buggar i olika system söker i större omfattning efter information av mer detaljerad art i den externa systemdokumentationen. Det skall gå snabbt att förstå och hitta rätt i systemet och den gruppen systemutvecklare har höga krav på att systemdokumentationen är tillräckligt detaljerad. En annan kategori är de systemutvecklare som främst ägnar sig åt nyutveckling och endast i enstaka fall vidareutvecklar existerande system. Dessa föredrar, i större utsträckning, att hämta alla detaljer direkt från källkoden.

Det anses dock av flera systemutvecklare vara mer effektivt och mer naturligt att använda sig av muntlig information än skriven dokumentation.

> *…systemdokumentationen förs över i någon slags muntlig tradition. Det är ju närmast oändligt mycket mer effektivt.*

Och många gånger, när systemdokumentationen helt saknas, är otillräcklig eller felaktig, är muntlig information den enda hjälp man kan få. Det påpekas dock att i praktiken är det vanskligt att förlita sig för mycket på muntlig information eftersom

flera företag i min studie har hög omsättning av personal och snabba projekt och har således inte så hög kontinuitet.

> *…det man lever på mycket är ju muntlig information från de som varit med i projektet. Och det är ju en väldig osäkerhet i att göra så, i och med att folk slutar och blir uppsagda då och då.*

De intervjuade systemutvecklarna använder den externa systemdokumentationen men inte i någon större omfattning. Den externa systemdokumentationen används för att *”komma igång och få en första känsla”* för systemet, men överges sedan för andra informationskällor, till exempel muntlig information från tidigare projektmedlemmar och framför allt källkoden.

5.2 Nyttan av systemdokumentation

Den verkliga nyttan av systemdokumentationen upplevs av flera systemutvecklare som liten i förhållande till den omfattning som den existerar och framför allt i förhållande till det arbete som krävs för att ta fram dokumentationen.

> *[System]dokumentationen ger inte tillräckligt mycket …[och] man får inte ut den nyttan som det kostar att ta fram den [systemdokumentationen].*

Jag vill dock betona att denna åsikt gäller enbart den externa systemdokumentationen och inte inline-dokumentationen där systemutvecklarna lättare kan se nyttan. Flera av systemutvecklarna upplever att systemdokumentationen inte är anpassad till deras behov av information vid t.ex. förvaltning.

> *Den [systemdokumentationen] är gjord med nån annan utgångspunkt. Det gör att jag kanske kan ha användning av den, det som jag letar efter kanske finns.*

En av anledningarna till att den externa dokumentationen inte upplevs vara till nytta är att det sällan finns index i tunga, informationsrika dokument. Och eftersom systemdokumentationen ofta används som ett uppslagsverk, få gånger läses ett dokument från början till slut, så blir det svårt att snabbt hitta den information man är ute efter. Vikten av index betonas starkt av de systemutvecklare som huvudsakligen förvaltar och vidareutvecklar existerande system. Så här sammanfattar en intervjuperson de viktigaste egenskaperna hos god systemdokumentation: *”mer lättläst, indexerad, [lätt att] slå och söka i”*.

Systemutvecklarna saknar också viktig information i den externa dokumentationen; *”det man verkligen behöver saknas ofta”*. Ofta saknas information om hur utvecklingsmiljön ser ut. Utvecklarna skulle gärna se att dokumentationen innehöll en detaljerad steg-för-steg instruktion för hur man installerar och konfigurerar utvecklingsmiljön. En annan information som nämns är information om vilka leveranser som gjorts såväl till kunden som till drift- och förvaltningsorganisation. Detta kan bidra till att *”man kan se nån slags spårbarhet”*, och upplevs som viktigt för de som satts att förvalta systemet.

> *…enkel versionshantering av källkoden finns ju men …vad som saknas är ordentlig konfigurationshantering för att ta reda på vad som har levererats.*

Dessutom så vill utvecklarna ha en komplett kravspecifikation med både funktionella och icke-funktionella krav. Alla utvecklare jag talat med anser detta vara fundamental information men upplever att det ändå ofta saknas.

Det råder lite olika meningar kring nyttan av databasdokumentation. Några systemutvecklare saknar databasdokumentation och ser gärna till exempel beskrivningar över stored procedures och vill även ha databasdiagram över hela databasen, speciellt när databaslösningen är icke-trivial. De upplever att det är enklare att ta till sig sådan information från en beskrivning i ett dokument än direkt från databasen; *”två, tre meningar om vad en stored procedure gör är ju mycket enklare än att gå in i databasen och läsa igenom all SQL kod”*. Andra utvecklare är av en motsatt uppfattning och anser att sådan information är överflödig i den externa dokumentationen och att det är enklare och säkrare att titta direkt i databasen. Generellt sätt så intar utvecklarna med många års erfarenhet den senare ståndpunkten.

Annan information som efterfrågas är brister och problem med systemet. Vad har man inte lyckats lösa tillfredställande? Vad hade man problem med under utvecklingen? Vad skulle kunna förbättras? Denna slags information anser flera systemutvecklare vara av stor hjälp och att det bör dokumenteras i den externa dokumentationen såväl som direkt i källkoden.

> *…saker man inte är nöjd med i koden bör ju dokumenteras…det behöver man inte skämmas för.*

Flera av systemutvecklarna klagar också över vad de anser vara onödig information i systemdokumentationen. Bland denna information kan nämnas: stora databasdiagram över icke-komplexa databaslösningar och ingående textuella beskrivningar om varje komponent och dess ingående funktioner.

Ett annat problem som påverkar systemutvecklarnas upplevelse av nyttan är att de inte litar på att systemdokumentationen är korrekt, att den är uppdaterad. Ingen av de intervjuade systemutvecklarna litar på att dokumentationen är uppdaterad. *”Man kan aldrig lita på dokumentation, så är det ju faktiskt.”* Och några utvecklare anser detta vara ett så stort problem att systemdokumentationen upplevs som *”helt oanvändbar”* och många gånger lika gärna kan kasseras. Under ett utvecklingsprojekt är det ju oftast oundvikligt att kraven förändras och för att hinna leverera i tid prioriteras dokumentationen ned; *”dokumentationen hänger inte alltid med och man är dålig på att uppdatera den i slutet”*.
Trots detta upplever några utvecklare att systemdokumentationen ändå är till hjälp. Framför allt gäller detta systemutvecklare vars främsta arbetsuppgift är att förvalta och vidareutveckla system. Visst, de tillstår att man inte kan lita på att dokumentationen är uppdaterad men *”allting hjälper…och ju mer [information] desto bättre”*.

Några av utvecklarna vittnar om att det finns en dokumentationsstandard men att den inte fungerar eller efterlevs; *”det finns [en dokumentationsstandard] men jag tror inte att den efterlevs, eller jag vet att den inte efterlevs”*. I dessa fall så är det mer eller mindre upp till varje utvecklares eget godtycke att tillämpa och tolka standarden.

> *…det är ju bra grejer som görs men det är ju enskilda personer som har tagit fram det på eget initiativ. Inte enligt någon standard.*

Detta sägs bero på att organisationen underskattat vad som krävs för att implementera en fungerande dokumentationsstandard. Man har tagit fram *”en uppsättning dokumentmallar”* och beskrivningar om *”hur man skall arbeta som projektledare”* men inte lyckats driva det i hamn; att se till att utbilda personalen och se till att standarden efterlevs. *”Man har aldrig orkat lägga ner det arbete som hade behövts helt enkelt.”*

Dessutom så granskas dokumentation sällan och flera systemutvecklare efterfrågar regelbundna granskningar får att höja kvaliteten på den levererade systemdokumentationen.

> *Det är väldigt sällan nån som granskar [system]dokumentationen. Man får leverera det man tycker.*

5.3 Tillgänglighet till systemdokumentation

Många av de intervjuade personerna vittnade om att de inte alltid vet var de skall hitta den systemdokumentation de är ute efter. Och de efterfrågar klarare riktlinjer över hur och var den externa dokumentationen skall lagras och hur den skall hanteras.

> *Det finns inget enkelt och organiserat ställe att leta efter [dokumentationen på], den kan finnas var som helst i princip. När man jobbar med ett projekt har man en plats där man lägger sina dokument och så länge arbetet pågår så funkar detta…men sen så helt plötsligt så är ju arbetet slut och då är ju dokumentarean i det tillståndet som det var när man slutade arbeta. Det görs sällan någon uppsamling där [på dokumentarean] för att förädla det här [systemdokumentationen] då, och egentligen färdigställa det för nåt annat syfte än att använda det som arbetsdokument.*

Dessutom så försämrar irrelevanta och gamla dokument tillgängligheten till den viktiga dokumentationen. Ett problem som beskrivs är att när ett systemutvecklingsprojekt avslutas så lämnas inte relevant systemdokumentation över till de personer som skall förvalta systemet. Vid behov så måste de systemutvecklare som satts att förvalta systemet själv leta upp den nödvändiga systemdokumentationen. Och den dokumentationen har ofta blivit liggande på en server bland all övrig dokumentation som tagits fram under projektets gång. Detta kan röra sig om hundratals filer. Denna mängd av dokumentation gör att det är svårt att hitta den information man är ute efter - *”det ligger där någonstans i den mappen, som då har 7, 8, 9, 10 undermappar”* då ger man lätt upp innan man ens har börjat leta.

> *…det finns inga riktlinjer för hur dokumentationen skall hanteras efter projektet är klart - att det skall rensas och avslutas och så…*

Detta problem skildras främst av de systemutvecklare som arbetar på medelstora till stora företag där projekten är stora och där det sällan är samma personer som förvaltar systemet som varit med och utvecklat systemet. *”Det är inte tydligt hur man ska leverera [system]dokumentation [till förvaltningsorganisationen].”*

En utvecklare beskriver också hur dokumentationsstandarden som används inom organisationen inte har en tillräckligt klar distinktion gällande vad som är systemdokumentation och vad som är projektdokumentation.

> *Ett annat problem är att…[dokumentations]standarden skiljer inte på rena projektdokument och systemdokumentation.*

Dessutom så upplever några av systemutvecklarna att de inte har tillräcklig kunskap om den gällande dokumentationsstandarden och därför inte alltid vet var (i vilket dokument eller modell) viss information går att hitta eller inte alltid förstår dokumentationen. UML är den notation som oftast (med överväldigande majoritet) används vid modellering, men endast två av sex systemutvecklare hade gått en kurs i UML.

5.4 Inline-dokumentation

Alla systemutvecklare jag talat med lägger stor vikt vid inline-dokumentation; *”den är väldigt viktig”*. Inline-dokumentationen är i allmänhet mer uppskattad som informationskälla än extern dokumentation. Den externa systemdokumentationen används ofta till att få en snabb överblick över systemet, medan detaljkunskaper samlas in via källkoden varför inline-dokumentation spelar en mycket viktig roll. God inline-dokumentation är speciellt viktigt för de systemutvecklare som är systemförvaltare. Tiden är ibland knapp för att lösa de problem som dyker upp och då måste det gå fort att förstå källkoden. Citatet nedan är från en intervjuperson vars arbetsuppgift det är att snabbt lösa allehanda problem med system han inte själv varit med och utvecklat.

> *…kunden har ju krav på att ärenden skall gå fort att lösa och de kanske inte kan förstå att en ändring på en rad tog tre timmar när två timmar och 45 minuter egentligen gått åt till att förstå koden.*

Dock litar man inte på att inline-dokumentationen är korrekt till 100 %, men har ändå en större tillit till den jämfört med den externa systemdokumentationen. Många tror också att systemutvecklare i allmänhet är mer benägna att hålla inline-dokumentationen uppdaterad än övrig dokumentation. Om man gör en ändring i systemet *”då är man ju där [i källkoden]… så det finns ju nästan inget motstånd alls mot att göra det [uppdatera inline-dokumentationen]”*.

Få av de systemutvecklare jag talat med har kommit i kontakt med källkod som innehållit för mycket inline-dokumentation, utan de flesta skräckexempel som jag fått ta del av beskriver endast motsatsen, total avsaknad av kommentarer.

> *…helt okommenterad kod har man ju sett [skratt]…då brukar jag själv kommentera den när jag tragglar mig igenom för att förstå. Men när man gör så, kommenterar nån annans kod i efterhand så tenderar det ju att bli på fel nivå…mer typ detta händer här och nu görs det än mer intressant information som varför och tänk på detta…*

I de fall när man har hittat störande kommentarer har det handlat om kommentarer som motsagt koden, överflödiga kommentarer (som antagligen har underlättat för programmeraren under utvecklingens gång) som inte rensats bort när utvecklingen avslutats eller när självklara och triviala saker kommenteras ymnigt. Ett exempel,

som både är trivial och som motsäger källkoden, (som kanske inte är så mycket störande som det är skrattretande) gavs av en intervjuperson. Följande fynd i ett program har blivit till ett stående internt skämt på den aktuella avdelningen:

```
//Vänta i 2 sekunder
sleep(1000);
```

I allmänhet anses det vara viktigare att kommentarerna i källkoden redogör för varför? man gjort på ett visst sätt än att de beskriver hur? man har gjort. Hur man har gjort framgår ju av källkoden. Annan information som efterfrågas i inline-dokumentationen är ändringshistorik.

> *Det är ju otroligt värdefullt att man direkt kan se i koden om en specifik bugg har åtgärdats när, av vem och varför. Det går ju att använda Source Safe så klart men det är lite bökigt …och som utvecklare tittar man inte gärna i Source Safe när man kan titta i koden. Dessutom är de sällan nån som använder Source Safe till nåt annat än att checka in och ut filer…ingen skriver några kommentarer eller så…*

Flera av systemutvecklarna har erfarenheter kring att använda inline-dokumentation i standardiserad form (t.ex. JavaDoc kommentarer och kommentarer i xml-format för .Net). Ofta beskrivs det som en bra teknik med många möjligheter som man i framtiden skall använda sig mer av. Men det vittnas även om vad som händer om tekniken inte tillämpas på ett vettigt sätt. I ett specifikt fall beskriver intervjupersonen konsekvensen av att man blivit överambitiös och dokumenterat även enkla och självklara saker i syfte att få en komplett dokumentation.

> *…varje enkelt event i code behind-filen…, en rad kod bäddades in i en halv sida html, helt oläsligt, det var bara en massa gytter…Om det är väldigt enkla och trivial saker så finns det ju ingen anledning Det är inte tekniken i sig som är dålig utan man måste tillämpa den på rätt sätt så att det blir kvalitet. Den måste komma upp över en viss nivå för att den ska vara användbar. Annars är risken att den stör så mycket att man inte bryr sig om den alls.*

I allmänhet så upplevs kvaliteten i inline-dokumentationen som relativt god och informationen är ofta till god hjälp.

6 Diskussion

I detta kapitel förs en avslutande diskussion med utgångspunkt från de teman i det empiriska materialet som presenterades i föregående kapitel: användning av systemdokumentation, nyttan av systemdokumentation, tillgänglighet till systemdokumentation och inline-dokumentation.

6.1 Användning av systemdokumentation

Den spontana reaktionen från några av intervjuobjekten när jag frågat om användandet av systemdokumentation har varit *Vilken dokumentation? Det finns ju ingen dokumentation!* När sedan intervjun har fortskridit har det framkommit att det faktiskt existerar dokumentation i större utsträckning än vad de först givit sken utav, men att dess nytta som informationskälla upplevs som liten eller att den är svårtillgänglig och att de därför, kanske omedvetet, inte räknar med dess existens.

Studien visar att systemdokumentationen används, dock inte i samma omfattning som den existerar och framför allt inte i proportion till den tid det tagit att producera dokumentationen. Detta beror på att tillgängligheten till systemdokumentationen ofta är dålig och att nyttan av systemdokumentationen upplevs som liten. Dessutom är deras tilltro låg till att systemdokumentationen överhuvudtaget finns och till att den är korrekt. Detta bidrar naturligtvis till att vissa systemutvecklare inte bemödar sig att försöka hitta systemdokumentationen. En annan anledning som bidrar till att systemdokumentationen inte används i större omfattning, är att skriven systemdokumentation inte upplevs som en lika effektiv och naturlig informationskälla som muntlig information. Wyssocky (2003) tar upp detta dilemma och tillstår att det visst är mer effektivt att kommunicera verbalt (speciellt under de tidiga faserna i ett utvecklingsprojekt), men att det inte alltid är möjligt. Dessutom är verbal kommunikation inte stabil; tidigare projektmedlemmar är inte tillgängliga för frågor eller är inte villiga att svara på frågor, utvecklare lämnar företaget och kunskapen hos de systemutvecklare som byggde systemet bleknar och försvinner med tiden. Därför menar Wyssocky (2003) att det är ohållbart att förlita sig alltför mycket på muntlig information.

Den externa systemdokumentationen används främst för att få överblick och en helhetssyn över systemet. Detta styrker den allmänna anvisningen att den externa systemdokumentationen bör fokusera på översikt framför detaljer (Booch, 1994; McConnell, 1996). Det är svårt att få en snabb översikt över systemet genom att enbart läsa källkoden och dess inline-dokumentation, för detta krävs extern dokumentation i form av modeller och textuella beskrivningar. Exempel på modeller som kan hjälpa till att få en översikt över systemet är diagram och beskrivningar över systemets arkitektur, systemets kontext och användningsfall. Användningsfall, som används för att beskriva de funktionella kraven på systemet, har utvecklats för att kunna användas i kommunikation med kund och/eller användare, men lämpar sig lika bra för systemutvecklare för att få överblick över funktionerna i systemet. Den mer ingående kunskapen om systemet (dess design och implementation) väljer utvecklarna att till stor del inhämta direkt via källkoden. Seamans studie (2002) visar på att systemförvaltarna förlitar sig till mycket stor del på källkoden och muntlig information. Min studie visar också på denna tendens men den är dock inte lika framträdande som i studien ovan.

Systemutvecklarna använder dock den externa systemdokumentationen för att få tillgång till information av mer detaljerad art. Detta gäller framför allt specifik driftsinformation, t.ex. information om utvecklingsmiljön respektive driftsmiljön och hur man installerar systemet.

Det hävdas ofta att systemutvecklare inte tycker om att läsa dokumentation, på samma sätt som de inte tycker om att skriva dokumentation. Detta antagande fann jag inget stöd för i min studie. I de fall där systemdokumentationen finns och är av god kvalitet så uppskattas den i allmänhet av systemutvecklarna. Studien visar på att nyttjandet av systemdokumentationen inte är större p.g.a. att den ofta är svårtillgänglig och p.g.a. att nyttan med den upplevs som liten eftersom den inte lever upp till systemutvecklarnas krav på information.

6.2 Nyttan av systemdokumentation

Nyttan av systemdokumentationen beskrivs som liten. Ett av de största problemen, som jag ser det, är att systemdokumentationen inte är skriven med den tänka läsaren i fokus.

Den tilltänkta läsaren ej i fokus

Systemdokumentation av god kvalitet bör utgå från den tilltänkta läsarens behov av information och inte från systemets uppbyggnad eller utvecklingsarbetets framskridande. Många gånger är man som systemutvecklare inte medveten om att det kommer att finnas en framtida läsare utanför det aktuella utvecklingsprojektet. Den systemdokumentation som produceras under ett utvecklingsprojekt används först och främst som arbetsdokument där den har till främsta uppgift att stödja kommunikationen mellan projektmedlemmar och säkerställa kvalitet i det resulterade systemet. Under projektets gång kompenseras luckor och oklarheter i systemdokumentationen med muntlig kommunikation mellan projektmedlemmarna. Det viktiga steget att sedan bearbeta och förädla dessa arbetsdokument och skapa användbar systemdokumentation som lämpar sig för förvaltning och vidareutveckling, är ett steg som sällan tas. Informationen som projektmedlemmar under ett utvecklingsprojekt behöver och informationen som behövs av en utvecklare som satts att förvalta ett system är till viss del olika i sin natur (Ambler, 2001a; Sommerville, 1995). Detta anses av många vara självklart. Ändå så upplever systemutvecklarna att systemdokumentationen inte har så stor giltighet efter det att utvecklingsprojektet är avslutat och systemet gått till förvaltning.

Ett konkret exempel på att systemdokumentationen inte är anpassad för den tänka läsaren, d.v.s. den utvecklare som satts att förvalta systemet, är att det sällan finns index i tjocka, innehållsrika dokument. Detta upplevs som ett problem eftersom ett *”dokument ska kunna användas som ett uppslagsverk”* (Mathiassen et. al., 1998, s.356) och ett uppslagsverk utan index är i princip värdelöst (Sommerville, 1992). I vissa fall kan även en ordlista och en lista med förklaringar över använda förkortningar vara nödvändigt (Mathiassen et. al., 1998).

Annan viktig information som är nödvändig vid förvaltning och vidareutveckling, och som ofta saknas, är detaljerad dokumentation om utvecklingsmiljön respektive driftsmiljön och information om vilka konfigurationer som har levererats. Projektmedlemmarna i ett utvecklingsprojekt sitter inne med mycket viktig

information som för dem är självklar men som måste dokumenteras för att kunna bevaras som värdefull information för kommande systemutvecklare.

Ett annat exempel är brister och problem med systemet. Man kanske har varit tvungen att välja en ”mindre elegant” lösning på ett problem på grund av en egenhet i systemet eller kanske på grund av tidsbrist. Detta är viktig information vid vidareutveckling som bör finnas med i systemdokumentationen (McConnell, 1996; Sommerville, 1992).

Det ideala vore naturligtvis om någon från förvaltningsorganisationen eller liknande får inblick i och kan ställa krav på den systemdokumentation som produceras. På samma sätt som kunden ställer krav på och påverkar systemet och eventuell dokumentation bör förvaltaren kunna påverka innehållet i systemdokumentationen. All dokumentation bör genomgå någon slags kvalitetsgranskning (Sommerville, 1992) och det bästa vore om systemdokumentationen kunde granskas av någon med god insikt i förvaltningsarbetet. Detta bidrar till att den producerade systemdokumentationen blir av bättre kvalitet samtidigt som medvetenheten ökar angående vilket innehåll och struktur systemdokumentation som lämpar sig för förvaltning bör ha. Mycket av den systemdokumentation som tas fram under utveckling av ett system syftar ju till att underlätta förvaltning och eventuell vidareutveckling. Däremot finns det en låg medvetenhet om hur användbar denna dokumentation verkligen är vid förvaltning och vidareutveckling (Seaman, 2002).

Vikten av att ha en dokumentationsstandard som fungerar och efterlevs kan inte nog understrykas. Flera av företagen i studien har någon slags dokumentationsstandard som definierar innehåll och struktur på dokument och modeller, vad Sommerville (1995) kallar *produktstandard*, men utvecklarna upplever att det är problem med tillämpningen eftersom standarden är dåligt förankrad i organisationen och att ingen ser till att standarden efterlevs. Dessutom så bör dokumentationsstandarden vara anpassad för alla olika roller och tilltänka användare som en viss typ av dokumentation kan tänkas ha. En dokumentationsstandard som enbart fokuserar på systemdokumentationens funktion under utvecklingsprojektet (och dess betydelse för det resulterande systemets kvalitet och projektets framgång) och inte beaktar behovet av information under ett systems förvaltningsfas är således inte tillräcklig.

Ej uppdaterad systemdokumentation

En annan bidragande orsak till varför nyttan av systemdokumentationen upplevs som liten är att systemutvecklarna upplever att systemdokumentationen sällan är uppdaterad. För att systemutvecklarna över huvud taget skall använda systemdokumentationen krävs att de kan lita på att informationen i den är korrekt. I motsats till Forwards studie (2002), som visade på att systemutvecklare och andra *”software professionals”* (s. 205) till viss del accepterar icke uppdaterad dokumentation, så visade inte min studie på liknande stor acceptans för dokumentation som inte överensstämmer med systemets implementation. Att utvecklarna inte litar på informationen i systemdokumentationen är en bidragande orsak till att nyttan upplevs som liten.

Inför ett utvecklingsprojekt är det lätt att man underskattar arbetsinsatsen för att skapa systemdokumentation av hög kvalitet. Ett annat problem kan vara att man inte inkluderar leverans av systemet och tillhörande dokumentation till förvaltning som

en del av utvecklingsprojektet (det kan vara svårt att motivera den kostnaden för kund och således räknas det inte med i budget) och att detta viktiga moment därför rinner ut i sanden. När tiden och pengarna är slut så lämnas projektet hals-över-huvud och dokumentationen lämnas således i ett icke färdigställt tillstånd och de tilltänka användarna får inte tillfälle att godkänna dokumentationen.
De flesta utvecklingsprojekt upplever nog tidspress av något slag i slutfasen av projektet. Och då är det naturligt att man prioriterar leverans av ett fungerande system framför att färdigställa systemdokumentationen (om inte kunden kräver denna dokumentation som en del av leveransen). Under ett utvecklingsprojekt finns det heller inget egenvärde i att ha dokumentationen helt uppdaterad vid alla tillfällen; det bidrar varken till att projektet blir klart snabbare eller till att systemet blir bättre (Ambler, 2001a). Det krävs därför ytterligare arbete för att färdigställa systemdokumentationen i slutet av projektet och göra den tillgänglig för förvaltning och att en tydlig leverans av systemet med tillhörande systemdokumentation görs till dem som skall förvalta systemet.

Problemet med att systemdokumentationen inte är färdigställd och anpassad för förvaltningsuppgiften när ett utvecklingsprojekt avslutas, grundar sig i en brist i planering och avsaknad av en mogen utvecklingsprocess i organisationen. Finns det en utvecklingsprocess och/eller en dokumentationsprocess måste den förankras i organisationen och organisationen måste följa upp och se till att processen efterlevs. Leverans av system med tillhörande systemdokumentation måste ses som en del i utvecklingsprojektet och således budgeteras och planeras för.

Flera systemutvecklare beskriver att det visst produceras systemdokumentation av god kvalitet men att det snarare beror på individuella personers engagemang, välvilja och kompetens än att det finns en standard eller process som efterlevs och fungerar. En parallell kan således dras med CMM och DMMs nivå 1. Nivå 1 kännetecknas av att projektets framgång är beroende av individuella hjälteinsatser i brist på en fungerande utvecklings- respektive dokumentationsprocess (Huang & Tilley, 2003; SEI, 2003).

6.3 Tillgängligheten till systemdokumentation

Den empiriska studien visar att det sällan finns en tillräckligt klar struktur över hur och var den externa systemdokumentationen lagras. Flera systemutvecklare vet inte alltid var systemdokumentationen finns. Detta indikerar att det finns en brist i hanteringen av systemdokumentationen. Det behövs klara riktlinjer för var och hur dokumentationen skall lagras, namngivningsregler, regler för versionshantering och eventuell arkivering av dokument.

Dåligt efterlevd dokumentationsstandard

De flesta företagen har riktlinjer för var systemdokumentationen lagras men de är antingen inte tillräckligt klara eller så efterlevs de inte. Det finns inget egenvärde i att ha en standard för dokumentationshantering när den inte efterlevs eller när den inte är tillräckligt förankrad i organisationen.

Dokumentationsstandard som inte skiljer på projekt- och produktdokumentation

Ett annat problem som beskrevs var att dokumentstandarden inte skiljer på projektdokumentation och produktdokumentation och att vikten av att ha denna distinktion inte är känd i organisationen. I allmänhet så kan man säga att rent projektrelaterad dokumentation har litet värde efter det att systemet har levererats (Sommerville, 1992). I de fall där systemdokumentationen skall ligga kvar på samma filarea som utvecklingsprojektet bör projektdokumentationen således antingen arkiveras eller tas bort. Systemdokumentationen får inte föra en tynande tillvaro på en server någonstans bland mängder av gamla irrelevanta projektdokument som skymmer den viktiga informationen. En annan lösning är att fysiskt skilja på projekt- och produktdokumentation redan från början. Detta kan bidra till ökad förståelse för den fundamentala skillnaden mellan *produkt* och *projekt*, och man blir således inte lika benägen att inkludera projektspecifik information i produktdokumentationen och vice versa.

Bristande kunskap om gällande dokumentationsstandard och notation

En annan aspekt av tillgänglighet till systemdokumentationen (som endast beskrevs av ett fåtal utvecklare men som jag ändå anser vara av vikt i diskussionen) är att det finns bristande kunskap om dokumentstandard och modelleringsnotation för att effektivt kunna ta till sig informationen i systemdokumentationen. Om man inte har tillräcklig kunskap om den gällande dokumentationsstandarden vet man inte i vilket dokument man skall leta efter den information man är ute efter. Om man dessutom inte har tillräckliga kunskaper i notationen som används, t.ex. UML, går mycket av informationen i diagrammen förlorad. Anmärkningsvärt var att endast två av utvecklarna hade gått en kurs i företagets notationsspråk (UML) och att ingen av utvecklarna (som arbetar på ett av de företagen som säger sig arbeta enligt en utvecklingsprocess och har en klar dokumentationsstandard) har genomgått en specifik utbildning med hänsyn till processen och/eller standarden.

6.4 Inline-dokumentation

Vid alla intervjuer jag gjort har inline-dokumentationen glömts bort, trots att jag innan varje intervju underströk att systemdokumentation lika mycket innefattar kommentarer i källkoden som externa dokument och modeller. Inte någon av systemutvecklarna har spontant talat om inline-dokumentation när de svarat på mina frågor angående systemdokumentation. Därför var det nödvändigt för mig att ta upp en diskussion om inline-dokumentationen separat i slutet på varje intervju. Detta har oftast blivit en lång och intressant diskussion. Systemutvecklarna saknar inte åsikter om inline-dokumentation, tvärt om, men är ovana att tänka på kommentarer i källkoden som systemdokumentation. Detta, tror jag, håller sakta på att ändras när fler och fler börjar använda mer standardiserad inline-dokumentation i syfte att exempelvis automatgenerera extern dokumentation i html format.

God inline-dokumentation är uppskattad av systemutvecklare och anses viktigt för förståelse av källkoden. Dessutom har utvecklare i allmänhet större tilltro till informationen i inline-kommentarer än i externa dokument och modeller. Trots detta upplever jag att det sällan diskuteras i det dagliga arbetet. Det finns sällan klara riktlinjer inom en organisation kring inline-dokumentationen motsvarande de som

finns gällande den externa dokumentationen. Detta kan ha många orsaker. Dokumentation i källkoden är ju väldigt ”kodnära” och kan därför vara ett känsligt område att styra alltför hårt, och många systemutvecklare motsätter sig kanske en sådan styrning. Dessutom så finns det i allmänhet bland ledningen i en organisation liten insikt i nyttan av inline-dokumentation och vad den består i; inline-dokumentationen är inte synlig för andra grupper än systemutvecklare.

Det finns dock problem och brister även med inline-dokumentationen; kvaliteten är varierande och kommentarer som är till för att öka förståelsen för källkoden kan ibland få motsatt effekt. I fallet med inline-dokumentation tror jag inte på en allmän standardisering liknande den för extern systemdokumentation. Jag tror i stället på en öppen diskussion mellan systemutvecklare om inline-dokumentationens syfte och nytta. Medvetenheten är ju olika hög hos olika systemutvecklare och en öppen diskussion i kombination med kontinuerlig kodgranskning, gärna i form av informella så kallade ”peer reviews”, kan bidra till att inline-dokumentationen över lag får en högre kvalitet. Kodgranskning bidrar ju även till en bättre kvalitet på källkoden och således även på systemet. McConnell (1996) hänvisar till studier som visar att dubbelt så många fel per timma hittas vid kodgranskning som vid vanlig testning och Humphrey (1997) menar att mellan 50 % och 70 % av alla fel i ett program hittas med hjälp av kodgranskning. Även Wyssocky (2003) rekommenderar någon typ av kodgranskningsprocess för att höja kvaliteten i källkoden såväl som i inline-kommentarerna. Jag anser också att det är viktigt att man har en debatt kring tekniken med standardiserade kommentarer i syfte att automatgenerera dokumentation för ett system, t.ex. de som finns för Java och .Net. Syftet och nyttan med denna teknik och när och hur tekniken är applicerbar bör diskuteras. Risken finns annars att tekniken missbrukas (som beskrevs av en intervjuperson) och att denna teknik används enbart för att den är tillgänglig.

7 Slutsats

I detta kapitel redogörs för slutsats och utvärdering av studien. Dessutom så ges uppslag till eventuella vidare studier inom problemområdet.

Systemdokumentation är ett evigt omtvistat ämne i IT-branschen. De flesta systemutvecklare har väldigt starka och bestämda åsikter om systemdokumentation, dess vara eller icke vara, kvalitet eller brist på kvalitet. Det var därför sällan svårt att få intervjuobjekten att tala men desto svårare att få dem att hålla sig till ämnet och den fråga jag ställt. Jag fick uppfattningen att många systemutvecklare har liten vana vid att diskutera användandet av systemdokumentation. Det är då lättare att tala om, eller snarare klaga på, bristen på systemdokumentation eller att redogöra för sina egna tillkortakommanden när det gäller att producera systemdokumentation.

Jag vill i detta kapitel återkoppla till frågorna jag ställde i inledningen och som jag hade som grund för min studie:
På vilket sätt använder systemutvecklare systemdokumentation som informationskälla vid förvaltning och vidareutveckling av system?
Hur uppfattar systemutvecklarna nyttan av systemdokumentationen?

7.1 På vilket sätt använder systemutvecklarna systemdokumentationen som informationskälla vid förvaltning och vidareutveckling av system?

Systemutvecklarna använder den externa systemdokumentationen främst för att få överblick och en första ingång till ett okänt system, men överger den sedan för andra informationskällor såsom muntlig information och källkod. Inline-dokumentationen används för att få detaljkunskaper om systemet och för att öka förståelsen för källkoden.

Det finns flera anledningar till att den externa systemdokumentationen inte nyttjas i större omfattning. Bland dem kan nämnas följande:

- Nyttan av den externa systemdokumentationen upplevs som liten (se nedan för vidare diskussion).
- Tillgängligheten till den externa systemdokumentationen är dålig (se nedan för vidare diskussion).
- Det finns en låg tilltro till att det existerar någon systemdokumentation och till att den är korrekt, vilket leder till att man inte letar upp den.
- Utvecklarna föredrar information via muntlig kommunikation eller den information som källkoden kan erbjuda.

7.2 Hur uppfattar systemutvecklarna nyttan av systemdokumentationen?

Den upplevda nyttan av systemdokumentationen beskrivs som liten, vilket främst beror på följande orsaker:

- Systemdokumentationen till form och innehåll är inte anpassad för den tilltänka läsaren; systemutvecklare som satts att förvalta eller vidareutveckla systemet. De som skall förvalta systemet har haft liten möjlighet att påverka systemdokumentationen till innehåll och form. Dessutom hinner man ofta inte färdigställa systemdokumentationen för den nya uppgiften p.g.a. bristande planering.

- Systemdokumentationen stämmer ofta inte överens med systemets implementation. Det finns inte tid i slutet av projektet att uppdatera och färdigställa systemdokumentationen.
- Tillgängligheten till den externa systemdokumentationen är ofta dålig. Detta beror dels på att de standarder som rör dokumenthanteringen (var och hur dokumenten lagras) inte efterlevs och dels på att utvecklingsprojekt inte avslutas på rätt sätt. Det är också vanligt att man inte skiljer på projekt- och produktdokumentation, varken fysiskt eller konceptuellt.
- Bristande kunskap hos systemutvecklarna om den gällande dokumentationsstandarden och modelleringsnotationen.

7.3 Hur öka nyttan och användningen av systemdokumentationen?

Med stöd av mitt empiriska material har jag kommit fram till ett antal konkreta riktlinjer som kan förbättra nyttan av och tillgängligheten till systemdokumentationen. Dessa sammanfattas nedan:

- En organisation bör se till att förankra den gällande dokumentationsstandarden och se till att den efterlevs. Detta inkluderar eventuell utbildning i utvecklingsmetod/-process, dokumentationsstandard/-process och notation.
- En bra dokumentationsstandard gör klara distinktioner mellan vad som är projekt- respektive produktdokumentation (systemdokumentation) både rent fysiskt (hur dokumentationen lagras) och konceptuellt.
- En bra dokumentationsstandard beaktar systemdokumentationens olika syften och tilltänka läsare, och fokuserar inte enbart på systemdokumentationens funktion under utvecklingen av systemet.
- De tilltänka användarna av dokumentationen (förvaltarna av systemet) bör få tillfälle att kravställa systemdokumentationen till innehåll och struktur och systemdokumentationen bör kvalitetsgranskas, gärna av någon med god insikt i förvaltningsarbetet.
- Leveransprocessen till förvaltning behöver tydliggöras. Utvecklingsprocessen som används skall tydligt inkludera leverans (av systemet och/eller systemdokumentationen) till förvaltning som en del av utvecklingsprojektet.
- Jag föreslår en öppen diskussion om inline-dokumentationens syfte och nytta i kombination med kodgranskning, t.ex. i form av ”peer reviews”, för att höja kvaliteten på inline-kommentarerna.

7.4 Utvärdering av studie

Så här i efterhand, med facit i hand, hade jag velat göra fler intervjuer med systemutvecklare vars främsta arbetsuppgift är att förvalta system. Jag tror att det skulle ha bidragit till ett mer komplett empiriskt material och givit mycket matnyttig information om brister och kvaliteter i den existerande systemdokumentationen. I alla slags kvalitativa intervjusammanhang finns det en risk att intervjupersonerna inte är helt sanningsenliga när de återger arbetssätt och vanor, detta kanske i syfte att framstå som mer kompetent och medveten i sin yrkesroll. Det finns även en medvetenhet om vad som anses vara ”politiskt korrekt” inom systemutveckling. De flesta vet hur man ”borde” göra och kan därför frestas att försköna sina utlägg om det egna arbetssättet. Det finns också en risk att jag som systemutvecklare, med mina egna erfarenheter och uppfattningar kring problemområdet, har färgat intervjufrågorna eller påverkat intervjupersonerna under intervjutillfället.

7.5 Uppslag till fortsatta studier

Under arbetets gång stötte jag på ett antal områden som jag personligen inte har så mycket kunskap om men som fångade mitt intresse. Ett exempel är Literate Programming, som det visserligen finns en del skrivet om internationellt, men bland systemutvecklare i Sverige är det ett relativt okänt begrepp. Det skulle vara intressant att undersöka om och hur Literate Programming tillämpas i Sverige idag. Agile är ett annat område (som i motsats till Literate Programming är välkänt) jag skulle vilja undersöka närmare. Framför allt då rent praktiskt, hur Agile-idéer tillämpas i utvecklingsprojekt.

8 Referenser

Böcker

Andersson, B-E. (1985). *Som man frågar får man svar - en introduktion i intervju- och enkätteknik*. Stockholm: Raben & Sjögren.

Avison, D. & Fitzgerald, G. (1995). *Information Systems Development: Methodologies, Techniques and Tools* (2nd edition). Maidenhead: McGraw-Hill.

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

Beck, K. (2000). *Extreme Programming Explained*. Reading MA: Addison-Wesley.

Booch, G. (1994). *Object-Oriented Analysis and Design With Applications* (2nd edition). Menlo Park CA: Addison-Wesley.

Brown, D. (1997). *An Introduction to Object-Oriented Analysis – Objects in Plain English*. New York NY: Wiley.

Eriksson, H-E. & Penker, M. (1998). *UML Toolkit*. New York: Wiley.

Humphrey, W., S. (1997). *Introduction to the Personal Software Process*. Reading MA: Addison-Wesley.

Jia, X. (2000). *Object-Oriented Software Development Using Java – principles, patterns and frameworks*. Reading MA: Addison-Wesley.

Kruchten, P. (2002). *The Rational Unified Process - En introduktion, Svenska utgåvan* (Susanne Dyrhage övers.). Boston MA: Addison-Wesley (Originalarbete publicerat 2000).

Mathiassen, L., Munk-Madsen, A., Nielsen, P., A. & Stage, J. (1998). *Objektorienterad analys och design*. Lund: Studentlitteratur.

McConnell, S. (1993). *Code Complete*. Redmond WA: Microsoft Press.

McConnell, S. (1996). *Rapid Development – Taming Wild Software Schedules*. Redmond WA: Microsoft Press.

Nickerson, R., C. (2001). *Business and Information Systems* (2nd edition). Upper Saddle River NJ: Prentice-Hall.

Sommerville, I. (1995). *Software Engineering* (5th edition). Harlow: Addison-Wesley.

Sommerville, I. (1992). *Software Engineering* (4th edition). Harlow: Addison-Wesley.

Strauss, A. & Corbin, J. (1998). *Basic Qualitative Research* (2nd edition). Thousand Oaks CA: Sage Publications.

Trauth, E. (2001). *Qualitative Research in IS: Issues and Trends*. Hershey PA: Idea Group Publishing.

Artiklar

Huang, S. & Tilley, S. (2003). Towards a Documentation Maturity Model. *Proceedings of the 21st annual International Conference on Documentation* (pp. 93-99). New York: ACM Press.
http://portal.acm.org [2003-12-18].

Paulk, M., S. (1994). *A Comparison of ISO 9001 and the Capability Maturity Model for Software*. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.
http://www.sei.cmu.edu/publications/documents/94.reports/94.tr.012.html [2003-12-18].

Seaman, C., B. (2002). The Information Gathering Stategies of Software Maintainers. *Proceedings of the International Conference on Software Maintenance*, October 2002 (pp. 141-9).
http://www.research.umbc.edu/~cseaman/papers/ICSM02.pdf [2002-01-04].

Uppsatser

Alexandersson, M. & Johansson, A-B. (2002). *Att dokumentera vid systemutveckling – en arbetsuppgift sedd ur systemutvecklarens perspektiv*. Göteborg: Göteborgs Universitet, Institutionen för informatik.
http://www.handels.gu.se/epc/ [2003-12-16].

Forward, A. (2002). *Software Documentation – Building and Maintaining Artefacts of Communication*. Ottawa: University of Ottawa, Ottawa-Carleton Institute for Computer Science.
http://www.site.uottawa.ca/~tcl/gradtheses/aforward/ [2003-12-11].

Smith, M. (2001). *Towards Modern Literate Programming*. Christchurch: University of Canterbury, Department of Computer Science.
http://www.cosc.Canterbury.ac.nz/research/reports/HonsReps/2001/hons_0110.pdf [2003-12-10].

Webbsidor

Ambler, S., W. (2001a). *Agile Documentation*, The Official Agile Modeling (AM) Site. [www dokument] URL
http://www.agilemodeling.com/essays/agileDocumentation.htm [2003-12-09].

Ambler, S., W. (2001b). *Agile Modeling and the Unified Process*, The Official Agile Modeling (AM) Site. [www dokument] URL
http://www.agilemodeling.com/essays/agileModelingRUP.htm [2003-12-11].

Borland. (2003). *Borland Together*, Borland Nordic site. [www dokument] URL http://www.borland.se/together/index.html [2003-12-28].

BSI. (2003). *Welcome to TickIT*, TickIT site. [www dokument] URL http://www.tickit.org/index.htm [2003-12-18].

Highsmith, J. (2001). *History: The Agile Manifesto*, Agile Manifesto Site. [www dokument] URL http://www.agilemanifesto.org/history.html [2003-12-04].

IEEE. (2003). *IEEE Standards Online*. [www dokument] URL http://standards.ieee.org/catalog/olis/arch_se.html [2003-12-31].

ISO. (2003). *International Organization for Standardization.* [www dokument] URL http://www.iso.ch/iso/en/ISOOnline.frontpage [2003-12-31].

Novalys. *Visual Expert Source Code Documentation*. [www dokument] URL http://www.visual-expert.com/us/info/technical_doc.htm [2003-12-28].

Rational. *Rational Rose XDE Developer*, IBM Software site. [www dokument] URL http://www-306.ibm.com/software/awdtools/developer/rosexde/ [2003-12-28].

Seaman, C., B. (2003). *Information Sources for Software Maintenance*. [www dokument] URL http://research.umbc.edu/~cseaman/maintenance.htm [2004-01-04].

SEI (Software Engineering Institute). (2003). *Capability Maturity Model for Software (SW-CMM)*, Carnegie Mellon Software Engineering Institute Site. [www dokument] URL http://www.sei.cmu.edu/cmm/ [2003-12-18].

TietoEnator. (2003). *TietoEnator PPS - Praktisk ProjektStyrning*. [www dokument] URL http://www2.tietoenator.com/pps/Default.htm [2003-12-22].

toolsfactory. (2003). *Doc-O-Matic – Source Code Documentation System*, Toolsfactory site. [www dokument] URL http://www.doc-o-matic.com/index.html [2003-12-28].

Wyssocky, L. (2003). *Source Code Documentation*, QP Project site. [www dokument] URL http://www.qualityprogramming.org/implementation/SourceCodeDocumentation/SourceCodeDocumentation.html [2003-12-09].