
QA og utviklere

TDT4235 Programvarekvalitet og prosessforbedring

Gruppe 10

Per Christian Henden

Johan Erlend Janbu

Henry Nygaard Scarborough

Vetle Valebjørg

Innledning

Dette essayet søker å besvare problemstillingen

“Kvalitetssikringsavdelingen blir ofte sett på med stor mistro blant utviklerne, blant annet fordi man mener de dreper kreativiteten i bedriften. Hvordan kan utviklerne og kvalitetssikrer jobbe sammen på en måte som gjør både utviklere og kunder fornøyde?”

I denne sammenhengen er det naturlig å se på hva QA-avdelingen (Quality Assurance, kvalitetssikring) og utviklerne kan gjøre for å legge forholdene til rette for å jobbe godt sammen. Det er imidlertid viktig å innse at QA-avdelingen og utviklerne er en del av et større bilde, og vi tar dermed også for oss hva bedriftene og utdanningsinstitusjonene kan gjøre for å bidra i denne prosessen.

Hva kan QA gjøre?

Et sentralt poeng for å kunne arbeide sammen på en måte som gjør alle sider fornøyde, er å la hver enkelt få lov til å gjøre jobben sin. I mange tilfeller viser det seg at programvareutviklere føler at deres sfære blir invadert ved at QA-avdelingen omtrent vil skrive koden for dem. Dette er svært uheldig for forholdet mellom dem. Dette gjelder også for QA-avdelingens forhold til selskapets ledelse — QA må passe på at de i tilstrekkelig grad informerer ledelsen, slik at disse sitter på nok informasjon til å ta de riktige beslutningene (Kelly & Ashley 1993).

Kunnskap om programmerernes jobb er for QA-avdelingen et tve-egget sverd. På den ene siden er det lettere å sette seg inn i programmerernes problemstillinger når man kjenner deres arena, men på den andre siden kan det da fort bli til at kvalitetssikreren kjenner enkelte deler av denne arenaen bedre enn programmererne selv, og fremstår da som en som ønsker å ha autoritet over andres fagområde, dersom denne personen fra QA i stor grad prøver å kommunisere de for-

bedringer som blir synlige på bakgrunn av denne bedre kjennskapen. Det gjelder for de ansatte i QA-avdelingen å ha en bevisst tilnærming til dette, slik at de ikke forsøker å overstyre programmerne (Riley 2004). Det vil i mange tilfeller være slik at dersom man demokratisk avgjør hva som er beste løsning, vil dette nå dårligere løsninger enn dersom det som den beste innenfor området ville foreslått i utgangspunktet hadde blitt fulgt (Sauer, Jeffery, Land & Yetton 2000). I denne sammenheng betyr dette at det er en stor fordel om man kan skille personenes kompetanse fra hverandre, slik at de som sitter på de beste kunnskapene innen et bestemt område kan være "ekspert" i denne mindre delen av totalområdet, og at deres forslag blir vektlagt.

Tilnærmingen til QA er svært ulik fra bedrift til bedrift. Noe som går igjen er at velviljen fra utviklere er større dersom QA har et større fokus på å være til hjelp enn å være ledelsens forlengede arm som en kontrollinstans. En kvalitetssikrer må være bevisst på at formålet er å forbedre kvaliteten på bedriftens produkter, ikke å kontrollere hver enkelt utvikler. Man oppnår bedre resultater ved samarbeid med utviklerne enn ved å forsøke å tvinge dem.

Å innføre kvalitetssikringen inkrementelt er svært sentralt når det gjelder aksept av QA (Hayes-Davies 1993, Kelly & Ashley 1993). Før innføring er det sentralt å kartlegge nåværende praksis og innrette innføringen av QA slik at styrkene ved dagens system blir best mulig ivarettatt samtidig som forbedringer gjøres. Man bør begynne med de delene av QA som gir håndfaste fordeler raskt, slik at programvareutviklerne får øynene opp for at QA er nyttig, og blir dermed umiddelbart bedre innstilt på å følge QA til daglig og samarbeide med QA-avdelingen. QA-avdelingen bør introdusere nye prinsipper i et tilstrekkelig lavt tempo til at hver bolk kan innarbeides rimelig godt før man går videre til neste.

Ved stegvis innføring kan bitene av det nye systemet etterhvert komme så mye i konflikt med biter av det gamle at et paradigmeskifte nærmer seg. Da vil det lønne seg å ta en fullstendig transisjon til ny tenkemåte/handlemåte innen dette området, da det gamle systemet her motarbeider det nye på så mange punkter at ytterligere små inkrement vil kunne bli oppfattet som en degradering eller forvanskning, og dermed kunne bli motarbeidet av medarbeidere. Dette er en

ond sirkel, som bør unngås.

Når man når en viss total størrelse på standardene som skal følges, vil dette av seg selv kunne føre til failed audits (Riley 2004). Det går en grense på noen hundre sider, avhengig av den enkelte utvikleren, hvor det blir svært vanskelig å holde oversikten over standardenes innhold. Failed audits skaper naturlig nok en følelse av nederlag, og mange failed audits skaper negativitet i bedriften. For å motvirke dette, er det viktig at QA hjelper til med å få ting på rett spor også før, ikke bare under, audits. Det er også viktig å få med ros i tilbakemeldingene, ikke bare ris (Solie 2002).

Det er viktig å ha et fokus på individuell tilrettelegging fremfor å ha en felles form alle skal passe i. Bedrifter har ulike behov, og en fast standard kan ikke dekke alt. Dette betyr ikke at man skal se bort fra standarder — det finnes fordeler ved konformitet både i forbindelse med kommunikasjon, effektivitet og mulighet for å kunne ta over andres arbeid, og mangel på kunnskaper kan delvis kompenseres ved hjelp av standarder utviklet av folk med grundig kjennskap til fagområdet. Det er imidlertid en balansegang her, og noen bedrifter befinner seg under balansepunktet. I tillegg vil standarder man anvender men ikke forstår i mange tilfeller gjøre vondt verre. Her kan man løse mye ved å bruke retningslinjer fremfor regler der det er hensiktsmessig (Eklund 1996).

QA-avdelingens testing av programmer bør i så stor grad som mulig inkludere testing av ting utviklerne ikke har tenkt på. QA-avdelingen har lite egen kompetanse på dette området i den nåværende situasjonen, men det er ønskelig om de kunne ha en dypere forståelse av denne delen av arbeidet deres enn i dag (Murugesan 1994).

Det er viktig å innse at det som gir kunder er ikke nødvendigvis minst mulig antall feil i programvareleveransene alene, men total “customer satisfaction”. Det er derfor bedre å fokusere på “customer satisfaction” fremfor å finne flest mulig antall feil (Murugesan 1994). Det vil eksistere situasjoner der minst mulig antall feil er spesielt viktig for kunden og sånt sett det helt dominerende kriteriet for hvor fornøyd kunden blir, men det er hovedsaklig innen bestemte domener,

hvor dette kriteriet er godt kjent, som for eksempel romfart, kjernekraftverk, våpensystemer og sentrale deler av banksystemer. Det er også viktig å innse at antall feil generelt ikke påvirker “customer satisfaction” direkte. Det er summen av problemene feilene skaper som teller, samt andre faktorer som inngår i hvor fornøyd kunden er. Når det gjelder selve utføringen av testen, gjør mange bedrifter nå kun bruk av faste testprosedyrer produsert av utviklerne (eventuelt i fellesskap med andre). Dette bør nok utføres i de fleste tilfeller, men kan med fordel suppleres i hvert fall med brukstesting hvor QA-avdelingen setter seg ned og prøver å bruke programmet slik kunden kommer til å gjøre, og kreativ testing hvor man aktivt forsøker å forårsake systemkrasj.

Til det spesifikke prinsippet om at QA dreper kreativiteten i bedriften, er et mulig svar å innføre metoder for å øke kreativiteten i bedriften. For eksempel kan man gjøre bruk av Edward de Bono’s ”Six Thinking Hats“ metode (de Bono 2003), som går ut på at medlemmer i en gruppe tenker parallelt. Man har blant annet hatten som representerer ”kreativitet“, ”logisk positiv (fordeler)“, ”følelser“ og så videre. Ved å innføre slike teknikker, som ikke er intuitive å komme på, men likevel gir raske håndfaste resultater, kan QA-avdelingen bidra til at kreativiteten i bedriften øker. Det kan også bidra til en bedre forståelse av egen rolle dersom QA-avdelingen bruker denne teknikken, da hatten ”logisk negativ“ (ulemper, feil) ikke bør stå alene i et konstruktivt samarbeid — QA-avdelingen må forsøke å bidra på andre områder også, når målet er best mulig programvare.

Et annet vanlig ankepunkt mot QA er at det er for byråkratisk og reduserer produktiviteten. Noen bedrifter (for eksempel japanske Hitachi) har imidlertid et helt bevisst forhold til akkurat dette, med filosofien at det ikke er noe poeng i å produsere upålitelig programvare raskere (Kelly & Ashley 1993).

Hva kan utviklerne gjøre?

Hva kan så utviklerne selv gjøre? Siden et av de sentrale momentene her er å endre utviklernes egne holdninger, handler det mye om å være åpne for kvalitetssikring.

Dersom omgivelsene bekrefter deres bange anelser, blir imidlertid gamle holdninger gjenopprettet gjennom erfaring, så det kan være et poeng at man har et QA-opplegg som fungerer noenlunde bra før man går løs på holdningene til den enkelte utvikler (Solie 2002, Thomas, Hurley & Barnes 1996).

Dersom utviklerne aktivt gjør bruk av QA-avdelingen i arbeidet sitt, i stedet for at QA-avdelingen må ta alle initiativ, vil dette kunne føre til stor forbedring både av arbeidet og av forholdet mellom avdelingene. Når utviklerne ser et behov, og kan få det løst ved hjelp av QA, ser de nytten ved samarbeidet, og QA får respekt ved at de bes involvere seg i noe. På denne måten blir det hjelp fra QA-avdelingen istedenfor “invasjon”.

Kommunikasjon er svært viktig for oppfattelsen mennesker får av hverandre. Dermed er det muligheter for forbedring i at det finnes en “metadialog” — at man diskuterer hvordan kommunikasjonen og samarbeidet fungerer, og prøver å forbedre dette. På denne måten er det mulig å ta eventuelle konflikter når de oppstår, eller til og med før de blir en konflikt, og en skalering av konflikten kan lettere unngås. Samtidig øker forståelsen av hverandres posisjon, som vil igjen bidra til mindre gnisninger i fremtiden.

Det er primært to ting man må passe seg for i forbindelse med metadialog. Det første er at ved å observere en prosess, påvirker man prosessen. Man må være forsiktig med å la metadiologen få for store konsekvenser for arbeidsform og lignende, helst ved at den skilles helt fra annen kommunikasjon. I tillegg må man vokte seg vel for å la metadiologen vokse seg stor i omfang. Hvordan samarbeide er et tema det kan tenkes lenge over. Det er viktig at fokus holdes strikt på målene med prosjekter osv., og ikke på å “lære å samarbeide”. Ellers kan man lett ende opp med at samarbeidet blir ineffektivt fordi ulike parter er for interessert i å se på hvordan samarbeidet

fortsatt foregår, og på grunn av det kan “vinninga gå opp i spinninga”.

Utviklerne kan også nyttiggjøre seg QA-avdelingen i forbindelse med mer kreative gjøremål. For eksempel når en kravspesifikasjon skal utarbeides, og man arrangerer en ATAM-prosess (Architecture Tradeoff Analysis Method), som er en metode som kort fortalt går ut på å få de forskjellige interessentene til å diskutere gjennom hva som er viktig i prosjektet, og hvilke hovedløsninger som bør velges (Bass, Clements & Kazman 2003), så kan man benytte dem både i planlegging av og som deltakere i denne prosessen. Dette vil både gi utviklerne mulige verdifulle innspill i en tidlig fase, samt gi kvalitetssikrerne bedre mulighet til å vurdere hvordan kvalitetssikringen bør tilpasses dette spesifikke prosjektet.

Hva annet kan QA og utviklere gjøre sammen?

Det er kjent at sosial omgang bedrer kommunikasjonen mellom grupper. Hvis det legges til rette for at personell fra QA-avdelingen og utviklere kan ha andre kontaktpunkter enn de rent jobbt tekniske, kan dette føre til en mer dynamisk dialog og et mindre anstrengt forhold til hverandre. Dette kan begge parter ta initiativ til selv, eller bedriften kan gå inn for det og legge til rette for det.

Hva kan bedriften gjøre?

Bedriften ønsker selvfølgelig, satt på spissen, at den skal produsere programvare på et øyeblikk, uten kostnader og uten feil (Yourdon 1995). Dette gir en “fortere, billigere, bedre” holdning som fører til stadige forsøk på effektivisering. Dette kan fort inkludere mange kutt i bedriftens utgifter som det ikke finnes dekning for; folk må jobbe mer, med høyere krav, og får skylda når ting likevel sprekker. Hvis bedriften kan holde fokus på “raskt nok, billig nok, godt nok” i stedet (Yourdon 1995), blir det mer harmoni i bedriftskulturen, og både QA-avdelingen og utviklerne

får bedre arbeidsforhold, som gjør at resultatene blir bedre. Dette betyr selvfølgelig ikke at man skal slappe av i forbindelse med effektivisering, men ha et mer realistisk forhold til hvordan det fungerer — man kan ikke kutte utgifter etter ostehøvelprinsippet (jevnt over uten å se om kuttene er mulige) og så forvente at kvaliteten opprettholdes. Bedriftens ledelse må sørge for at spørsmål og konflikter vedrørende standarder og lignende går inn i en ryddig og oversiktlig prosess. Klager på standarder osv., som QA-avdelingen har tatt i bruk, må refereres dit, hvor standarden bør vurderes.

Dersom avdelingen kommer frem til at standarden bør være slik den er når man tar i betraktning den informasjon som klageren sitter på, må ledelsen støtte opp om dette i de fleste tilfeller, hvis QA-avdelingen skal få nevneverdig respekt blant utviklerne. Her er det selvfølgelig også viktig at det QA-avdelingen kommer frem til holder mål, hvis ikke ender man opp med å øke konfliktene og får ikke ordnet opp i de problemene som stammer fra QA-avdelingen.

Her er det også viktig at QA-avdelingen faktisk får innflytelse nok til å gjøre noe med problemer de ser innenfor sitt område (Runeson & Isacsson 1998). Ved å innføre en rutine om at de kan rapportere problemer til ledelsen, hvor rapporten faktisk blir fulgt opp, får man nyttiggjort seg deres kompetanse og samtidig oppmuntret dem ved at de ser at det nytter å rapportere problemer.

For å unngå problemet med at QA-avdelingen forsøker å gjøre programmerernes jobb, kan det være fordelaktig å ansette folk med erfaring fra QA innen andre bransjer, fremfor detaljkunnskaper innen egen bransje, da dette vil føre til en større forståelse fra begge sider om at både de selv og andre sitter på verdifull kompetanse som ikke andre parter har. Dette vil igjen føre til større respekt for hverandres fagfelt, og et mer konstruktivt samarbeide (Riley 2004).

Det er også et poeng å unngå at utviklerne gjør QA-avdelingens jobb. I noen bedrifter får utviklere prestasjonslønn som er utregnet slik at jo flere feil kvalitetssikrere finner i en utvikler sin kode, jo mindre lønn får utvikleren. Dette er svært uheldig, da utvikleren får et økonomisk incitament til å prøve å gjette hvilke tester QA-avdelingen vil utføre, og så utføre disse selv for å fjerne akkurat de feilene som uansett ville blitt funnet før produktet når ut til brukerne. Dette er

svært dårlig bruk av ressurser og bør derfor unngås.

Hva kan undervisningsinstitusjonene gjøre?

Studier innen datateknikk og informasjonsvitenskap kan bidra til bedre samarbeid mellom utviklere og QA ved å tydelig kommunisere moderne norsk bedriftskultur, blant annet det at man fokuserer på å være gode nok i alt, men i verdensklasse på en spesialitet. For å nå verdensklasse må alle virkemidler tas i bruk, deriblant mange forskjellige QA-grep. Det at de lærer dette og får en introduksjon i disse prinsippene gjennom fagstudiene, gjør at de er mer åpen og konstruktiv i forhold til det når man møter det i arbeidslivet.

Mye fokus på hvordan QA skal gjøres er til nå basert på at man er en stor organisasjon, hvor det er en egen QA-avdeling som tar seg av det. Det er gjort for lite forskning på hvordan dette fungerer i små bedrifter, og her trengs det mer forskning for å kunne gi kvalifisert rådgivning i hvordan små bedrifter kan forbedre seg (Kautz 1999). Det er også andre felt innen QA som er dårlig dekket, slik at man vet lite om hva man kan oppnå og hvordan det best oppnås.

Noe som i stor grad ville bidra, er om undervisningsinstitusjonene tilbyr et eget studium som retter seg utelukkende mot QA. Dette bør basere seg på de generelle prinsipper som kvalitetssikring bygger på som fundament, og deretter tilby spesialiseringer som påbygninger. På denne måten oppnås det et eget fagfelt hvor fokuset blir på den jobben QA-staben skal utføre i en bedrift. De ulike industrier kan dra nytte av erfaring på hverandres område, og situasjoner hvor kvalitetssikrerne tar andres jobb pga. at den jobben de sikrer er hva de selv er utdannet til, unngås. Innenfor et tilhørende fagmiljø vil det finnes gode muligheter for å forske videre på QA, der manglene er store.

Et slikt studium vil i seg selv føre til mer respekt for QA-avdelingens arbeid, da dette vil være et eget fagområde, som de ansatte der vil være fagpersoner i. Gjennom dette studiet kan de få mange innspill på hvordan man best kan arbeide med “produsentene” i bedriften, for å utvikle

samhandling til det beste for alle parter. Det har også den fordelen at dersom en industri eller et fagfelt har en dårlig periode med nedgang i antall arbeidsplasser, kan de som har QA-utdanning søke mot andre industrier med eller uten et lite påbygningskurs, noe som vil være til stor fordel både for de det gjelder og for samfunnsøkonomien generelt.

Innen et slikt studium vil det også være mulig å ta en spesialisering innen det å hjelpe småbedrifter som konsulent, dersom forskning viser at det er god økonomi i å hyre inn konsulenter for de mindre bedriftene.

Konklusjon

Det er mye som kan gjøres for å oppnå forbedringer med tanke på problemstillingen. Innføring av et eget QA-studie på tvers av industriene er noe som er lite beskrevet i nåværende litteratur, og ville være en spennende nyvinning som kan ha mange positive ringvirkninger, både på kommunikasjon, kunnskaper og samarbeidsforhold. Ikke minst vil det også gi kvalitetssikring status som eget fagfelt, som vil gjøre det langt lettere å oppnå konstruktive forhold mellom QA-avdelingen og andre avdelinger i bedrifter. Det gjelder også for QA å ha et bevisst fokus på hvordan samarbeidet fungerer, gjerne ved å introdusere en metadialog, ved å tilpasse kvalitetssikringen til hva som er hensiktsmessig i situasjonen og ved å introdusere teknikker for å motvirke negative aspekter andre mener QA har (hvorvidt disse er reelle eller ikke). De må videre være klar over egen rolle i bedriften. For utviklerne handler det mye om å inkludere og gjøre bruk av kvalitetssikrerne i sitt arbeid, for på den måten å få bedre resultater, bedre samarbeidsklime og vise at man verdsetter arbeidet QA-avdelingen gjør.

Referanser

- Bass, Clements & Kazman (2003), *Software Architecture in Practice*, 2. edn, Addison-Wesley Professional.
- de Bono, E. (2003), 'Six thinking hats — tools for parallel thinking', Training material. Advanced Practical Thinking Training, Inc.
- Eklund, B. (1996), 'Iso 9000 is no miracle cure', *Engineering Management Journal* **6**(4), pp. 169–171.
- Fresh Unpasteurized Apple Juice/Cider Quality Assurance Plan* (2001). Tilgjengelig på Apple Hill Juice/Cider Processors sine hjemmesider på http://www.co.el-dorado.ca.us/ag/apple_cider/qualityplan.html.
- Graham, D. R. (1993), 'Testing, verification and validation', *IEE Colloquium on Layman's Guide to Software Quality* .
- Hayes-Davies, P. (1993), 'Process maturity', *IEE Colloquium on Layman's Guide to Software Quality* .
- Kautz, K. H. (1999), 'Making sense of measurement for small organizations', *IEEE Software* **16**(2), pp. 14–20.
- Kelly & Ashley (1993), 'Measurement as a powerful management tool', *IEE Colloquium on Layman's Guide to Software Quality* .
- Murugesan, S. (1994), Attitude towards testing: A key contributor to software quality, in 'Software Testing, Reliability and Quality Assurance', pp. pp. 111–115.
- Parnas & Lawford (2003), 'The role of inspection in software quality assurance', *IEEE Transactions on Software Engineering* **29**(8), pp. 674–676.

- Riley, P. (2004). Intervju av Paul D. Riley (ISO auditor og programvareutvikler). Intervjuet ble gjort av Erlend Janbu, en av forfatterne av denne artikkelen, den 8. oktober.
- Runeson & Isacsson (1998), Software quality assurance — concepts and misconceptions, *in* 'EUROMICRO Workshop on Software Process and Product Improvement', pp. pp. 853–859.
- Sauer, Jeffery, Land & Yetton (2000), 'The effectiveness of software development technical reviews: a behaviourally motivated program of research', *IEEE Transactions on Software Engineering* .
- Solie, C. (2002), 'The myths of quality assurance'. Tilgjengelig på <http://www.pei-911.com/HTML/article-myths.pdf>.
- Thomas, Hurley & Barnes (1996), Looking for the human factors in software quality management, *in* 'International Conference on Software Engineering: Education & Practice'.
- Yourdon, E. (1995), 'When good enough software is best', *IEEE Software* **12**(3).