Automatiserad release som en del av var story

David Larsson, lak03dbo
&
Lars Gustafson, ada10lgu

5 mars 2013

Abstract

Continuous Deployment (CD) is a discipline with the core practice to create a working release of software as often as possible during development. Every time a new piece of functionality is added, the software is deployed. To accomplish this several things are required: automatic test suites, version control and a well-documented release process. This paper is about incorporating some level of CD into a student project using eXtreme Programming (XP), where every story would lead to a release. The aim of the experiment was to reduce stress levels, increase efficiency of the team and quality of the code. The results were inconclusive. The stress levels seemed to decrease, the efficiency of the team decreased or was unaltered, the quality of code seemed to increase, with one exception.

Innehåll

1 Introduktion

Agila mjukvaroprojekt har som motto att jobba högiterativt, det vill säga att bygga upp produkten successivt i omgångar. I kursen *coachning av programvaruteam* på LTH ges möjligheten att agera coacher för ett team av studenter som läser kursen *programvaruutveckling i grupp* (PVG). I sagda kurs jobbar teamet enligt eXtreme Programming-metoden (XP) som beskrivs något mer i detalj nedan.

Continuous Deployment (CD) är en metod som används inom XP-projekt och andra agila projekt [1]. Den går ut på att varje *story* leder till en ny release. I vanliga fall sker releaser inte lika ofta. Ett projekt kanske har ett fåtal delreleaser och därpå en slutrelease. Använder man CD sker flera releaser om dagen. Dessa releaser behöver inte alla levereras externt till kunden, utan kan levereras internt som release-kandidater. Vi har valt att försöka låta vårt PVG-team nytta CD i sitt arbete. Den främsta anledningen till detta är vår egen erfarenhet av att läsa PVG-kursen, samt diskussion med andra personer som läst kursen lett till en konsensus om att releaserna är de mest stressfyllda momenten i hela arbetet. I vår studie har vi inriktat oss på CD i kombination med XP och hur det kan hjälpa teamet att få en mer stressfri och effektiv utvecklarmiljö. Våra tre frågeställningar är:

- Kan man minska stressen vid en release genom att nyttja CD?
- Ökar produktiviteten i teamet genom att tidigt introducera CD?
- Ökar kodkvalitén och blir mjukvaran mer buggfri vid nyttjande av CD?

Den sista frågeställningen fanns inte initialt när idén kring studien kläcktes, men tillkom när arbetet fortgick.

2 Bakgrund

Nedan beskrivs de olika aspekter som är relevanta för rapporten men kräver ytterligare beskrivning för den oinvigde att förstå.

2.1 eXtreme Programming

Extreme programming [2] är en högiterativ agil utvecklingsmetod, vilken inriktar sig på att låta kunden vara en del av teamet vid utveckling av mjukvara. Vidare ligger fokus på att jobba i väldigt korta iterationer och nyttja ett antal *practices* och *values* som speglar inställningen man bör ha när man arbetar enligt XP-metoden. Så kallade stories är en stor del av XP. Kunden skriver *stories* med funktionalitet som han vill ha implementerad, teamet estimerar hur mycket de kommer kosta och kunden får sedan prioritera dem han helst vill ha med i nästa iteration. Detta sker på ett planeringsmöte. Arbetet fortlöper sedan med inställningen att förhållandena ständigt kan ändras och att man ska ta till sig och välkomna denna förändring. För att inhämta kunskaper som behövs för framtida utveckling nyttjas även *spikes*. En *spike* motsvarar tid att lära sig eller experimentera på egen hand.

2.2 Continuous Deployment

I många mjukvaruprojekt är releaseprocessen väldigt tidskrävande [3], riskfull [5] och jobbig för utvecklingsteamet. Det är ett stort stressmoment och kan ta timmar eller dagar. Boven i dramat är enligt J. Humble et. al [1] bland annat två så kallade *anti-patterns*:

- Att göra en release manuellt
- Att göra en release till kundmiljö först när produkten är klar

Att göra en release manuellt innefattar bland annat manuellt testande, ett långt dokument över vad som behöver göras inför varje release, samt tidsspillan när något går fel och releasen behöver göras om. Att göra en release till kundmiljö först när produkten är klar beskriver hur problem kan uppstå om systemet ej testas utanför den säkra utvecklingsmiljön genom utvecklingens gång. Detta leder till att delar av releasen skickas iväg otestade och kanske inte alls fungerar som det ska när de ska köras på en annan plattform. Vidare kan det leda till att allvarliga problem inte upptäcks förrän nära releasedagen [5].

CD (även continuous delivery) är enligt flera [3][4][5] lösningen på dessa problem. CD är en disciplin som innebär att varje tillagd funktionalitet i mjukvara blir en release. Releaseprocessen automatiseras så långt det är möjligt, processen avdramatiseras och omedelbar återkoppling ges i varje steg [1]. Man har myntat begreppet *The Deployment Pipeline*, vilket ska beskriva en flödessystem för utvecking, i vilket det sker check-ins till olika nivåer. Dessa nivåer inkorporerar olika tester som ger omedelbar feedback och triggar nästa nivå stegvis tills dess ett fel upptäcks eller en ny release har skapats (bild) [5]. I detta flödessystem ligger nyckeln till att lyckas med CD: automatisering och feedback. Genom att automatisera releaseprocessen och bygga in feedback upptäcks fel tidigt, då de är billigare att laga. Dessutom minskar stressnivåerna hos utvecklarna eftersom att releaseprocessen blir trivial [1]. Utöver detta så skapas en flexibilitet eftersom att teamet alltid är redo att göra en release om kunden skulle kräva det.

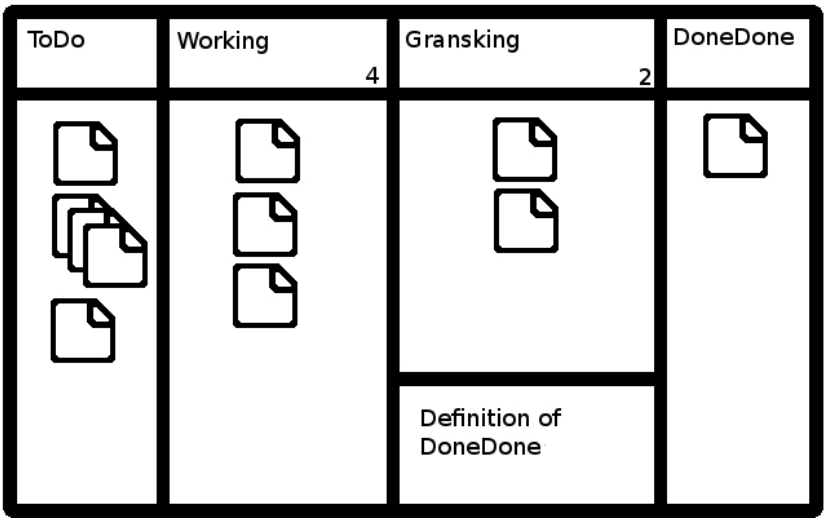
Ett begrepp som är intimt knutet till CD och används inom CD är Continuous Integration (CI). CI innebär att mjukvara under utveckling alltid måste vara i ett fungerande läge. Varje gång applikationen byggs så körs ett stort antal automatiserade tester. Om ett fel upptäcks så lagas det direkt.

2.2.1 Hur CD appliceras i studien

I vår studie har vi valt att applicera delar av CD i kombination med XP genom att introducera ett krav att varje story måste leda till en ny release. Flera av de delar som ingår i CD, såsom automatiserade tester och CI ingår redan som practices i XP, vilket gjorde att steget inte blev så stort att ta. Det skall dock tilläggas att eftersom att CD kräver helautomatiserad releaseprocess så tvingades vi låta teamet lära sig detta tidigt - redan under första iterationen började de med en specialstory att lära sig implementera ANT-script.

2.3 Kanban

Kanban är en metod som kommer ifrån det japanska bilföretaget Toyota [6][7], för att öka produktiviteten i fabrikerna. Ett kanbanbräde är en visuell presentation över hur arbetet fortlöper. På senare dagar har detta även tillämpats i agila metoder och vår tolkning kommer från en djupstudie av Niklas Fors och Niklas Hansson [8]. Vi har dock modifierat deras lösning så den passar vårt ändamål med CD. Vi lade in *Klar för release* i *DoneDone*, då vi gör en release efter var story, samt hade ingen uttalad checklista för att få flytta något från *utveckling* till *granskning*. Liksom Fors och Hansson hade vi även kvar att endast fyra stories får finnas på utveckling, eller *working* som vi kom att kalla det, samtidigt samt två på granskning (Figur 1. Vi var även noga på att poängtera vår tolkning av Kanban och dess användning. Nyckeln ligger i begränsningarna på mängden stories som får finnas i faserna working och granskning. Genom att ingen ny story får flyttas till granskning om där redan finns två så tvingar det utvecklarna att granska en story innan de börjar på en ny. Denna flaskhals ger ett flöde som garanterar att de stories som måste hinnas med blir klara.



Figur 1: En schematisk bild över kanbanbrädet.

2.4 Definition of DoneDone

Målet med DoneDone [8] är att ge gruppen en klar checklist för att säkerställa den kvallitet och funktionalitet vi lovat kunden. Hur listan skulle utformas från start var inte självklart utan vi valde istället att låta gruppen göra ett första utkast. Med deras lista som grund samt med nya moment som uppstått under projektets gång, så som manual, så har vi coacher vidareutvecklat listan. Som en liten extra kontroll lade vi även in tillfälliga delmoment så som att de måsta gå till coacherna och säga en specifik fras. Detta gjordes så att vi som coacher lätt kunde se att de läste listan noggrant.

Slutligen såg vår lista ut på följande sätt:

- All ny funktionallitet är implementerad

- Koden är kommenterad
- Inga kompileringsfel finns
- Alla felmedelanden skrivs ut med förklarande meddelanden och inga ss-tack-traces"
- Lyckad exekvering meddelas i sorteringsprogrammet (Endast sortering)
- Koden är testad och klarar testerna
- Manualen är uppdaterad så den speglar ny funktionallitet
- Uppdatera javadoc(alla publika metoder och klasser ska ha kommentarer)
- Filen med listade storys är uppdaterad
- ANT-scriptet omfattar allt
- En release till teamet är gjort
- Releasen är generellt testad med fokus på den akutella storyn

2.5 ANT-script

ANT-script [9] är ett Apache open source-verktyg för att automatisera byggandet av ett program från programkod. Det innehåller metoder för att kompilera kod, köra tester, bygga mappstruktur, generera körbara filer, packa ihop till ett arkiv med mera. Genom att nyttja XML-syntax skapas en enkel, taggbaserad kod som även kan innehålla beroenden mellan olika delar av det som byggs. Exempelvis är testning beroende av att kompilering har skett. För att kunna nyttja CD krävs en automatiserad releaseprocess [1]. ANT har därför använts i projektet för att snabbt kunna göra alla de saker som krävs för att producera en release, och därmed skala bort tiden det tar att utföra alla dessa operationer manuellt.

2.6 Kursupplägget

Projektet innehåller personer från två olika kurser. Själva utvecklarna i teamet består av medlemmar av kursen *Programvaruutveckling i grupp* (kurskod: eda260) medan de två coacher, författarna bakom denna rapport, går *Coaching av programvaruteam* (kurskod: eda270). Coacherna har vid tidigare tillfälle själv läst eda260 och har därmed en bra inblick i hur projektet går till samt vad som kommer hända. Coacherna ska inte enbart agera coacher utan även dubbla som mentorer och så kallade trackers, vilka har kontroll på hur teamet ligger till med sina stories utifrån vad som har estimerats att hinnas med.

2.7 Continuous Deployment i arbetslivet

Även om vi inte har haft möjlighet att undersöka det noggrannare så har vi funnit företag som arbetar med CD ute i industrin. ThoughtWorks [5] är ett multinationellt företag som nyttjar CD. Vidare har vi varit i kontakt med ett danskt företag som heter Praqma [10] vilka enligt egen utsago ska vara experter inom just CD.

3 Studien

Studien kopplad till denna rapport utfördes på ett team av studenter i årskurs 2 på datateknikprogrammet på Lunds Tekniska Högskola i Lund. Författarna agerade coacher och mentorer men utförde samtidigt en studie där de introducerade CD till sitt studentteam. Nedan beskrivs det som framkommit under studiens gång, samt ett resultat och en diskussion i ämnet.

3.1 Hypotes

Vår hypotes är att det initialt kommer bli kämpigt att lära sig automatisera releases och att de kommer hinna med färre vanliga stories än andra team. Efterhand hoppas vi dock att automatiseringen kommer att ge dem en säkerhet och trygghet i att skapa frekventa releaser. När så kunden kräver en extern release, kommer teamet helt enkelt kunna välja senast fungerande release och skicka denna, istället för att försöka få ihop en ny innan deadline. Anledningen till att det förmodligen kommer bli kämpigt i början är att CD kräver en hel del förarbete i form av att lära sig göra en release, automatisera denna, samt se till att programmet är körbart och innehåller alla filer som behövs. Till detta har vi planerat att låta teamet använda en kombination av kanbanbräde, automatiska tester, en väldefinierad DoneDone-lista och ANT-script.

3.2 Utvärderingar

För att på något sätt kunna mäta vårt resultat har vi utformat två enkäter och en intervju. Målet med den första enkäten var att få en generell bild över alla team och hur de fungerade. I ett senare skede intervjuade vi vårt team en och en för att få en bättre inblick i deras åsikter kring CD, samt fördelar och nackdelar med dess införande. Slutligen skickade vi även ut en kort enkät till alla de andra teamens (12st) coachpar. Meningen med denna enkät var att få in data över hur deras releaser hade gått jämfört med vårt teams med avseende på stabilitet och kodkvalitet. Enkäterna finns i sin helhet i Appendix A,B och C. Resultatet för respektive enkät beskrivs i resultatdelen av rapporten.

3.3 Det iterativa arbetet

Nedan följer en beskrivning om hur arbetet genom kursen fortlöpte vecka efter vecka. Varje måndag utfördes en iteration om åtta timmars utveckling. Dessa beskrivs i en något mer berättande stil än resten av rapporten. Fokus läggs på hur arbetet med CD och dess komponenter växte fram under arbetets gång, samt hur teamet reagerade och utvecklades.

3.3.1 Iteration 1

Vi gav teamet en extra story redan vid första planeringsmötet. Denna story behandlade automatiserade releases. Det blev ett stort steg för dessa unga, orutinerade programmerare att behöva sätta sig in i ANT-script och XML-syntax. Det gavs en spike till två personer att nysta i hur sådant fungerar och på kommande långlaboration fick hela tiden två personer sitta och jobba med scriptet. De kom en bra bit på väg men det skulle komma att dröja ytterligare ett

par iterationer innan resultaten började uppdaga sig. Vidare började DoneDone-listan ta form, dock utan att innehålla automatiserade releaser och CD-kravet att var story leder till en release.

3.3.2 Iteration 2

Under iteration 2 förväntade sig kunden en första release. Denna release var planerad att ta någon timme men drog ut på tiden rejält på grund av diverse problem med filer, sökvägar och icke-matchande manual. Här gavs inte utrymme att nyttja ANT-scriptets kraft fullt ut. Dock kunde vi coacher redan här se att de hade viss nytta av scriptet för att bygga själva releasen.

3.3.3 Iteration 3

I tredje iterationen var ANT-scriptet färdigt. Det kunde nu kompilera, bygga mappstruktur, köra alla tester, skapa körbar jar-fil, packa in såväl källkod som nödvändiga filer i ett arkiv, samt maila till kunden. Listan över kriterier för att en story skulle få räknas som helt färdig, DoneDone, utökades med att man skulle bygga en release som skickades internt till teamet.

En ny release till kunden var planerad under eftermiddagen. Även här hade teamet problem att leverera i tid, men problemet låg inte hos releasen som sådan, utan snarare i att all funktionalitet inte fanns på plats. Således kunde vi här se att teamet fått rejäl hjälp av sin automatiserade releaseprocess. Vi började ana ljuset i tunneln.

Genom att ha den här kontinuerliga releaseprocessen i varje story skapas en extra säkerhet hos teamet att produkten verkligen fungerar. Vem som helst i teamet kan när som helst testköra en release och undersöka dess funktionalitet, även utanför den skyddande atmosfär som utvecklingsverktyget Eclipse skapar.

Det ska dock tilläggas att införandet av CD också hade sitt pris. I och med att teamet under nästan tre hela iterationer i det här läget har fått avsätta två personer under hela iterationen till att arbeta med den automatiserade releaseprocessen så kunde enbart sex personer jobba med de vanliga stories som skulle hinnas med. Det märktes genom att teamet hamnade efter jämfört med andra team enligt kunden.

3.3.4 Iteration 4

Utvecklingsarbetet fortgick enligt planen. Teamet har lagt till i DoneDone-listan att en release måste produceras internt och testköras med inriktning på den aktuella storyns tillförda funktionalitet innan en story får flyttas från Granskning till DoneDone. Arbetet har fungerat väl och det stora nät av tester och granskning som har byggts upp gör att få buggar slipper genom till DoneDone. Det hände med jämna mellanrum att en story fick flyttas baklänges från Granskning till Working när granskarna upptäckte buggar eller att någon funktionalitet missats, men det anser vi vara en del av processen och ett extra skydd mot att buggar följer med i en release.

3.3.5 Iteration 5

Denna veckan hade gruppen en stor moraldipp. Produktionen blev lidande och vi som coacher var tämligen maktlösa att bryta mönstret. Med det i åtanke så lyckades teamet ändå producera en fullständig release och skickas till kund samt ett annat team för utvärdering. Kunden var nöjd och allt fungerande även om viss funktionalitet saknades. Moralen inom gruppen var dock hög - ingen gnällde på någon annan eller beskyllde dem med att ha gjort fel. Snarare gaddade teamet ihop sig mot kursen och omvärlden. De upplevde att kursen började uppta lite väl lång tid.

3.3.6 Iteration 6

Under den sista iterationen var det en annan stämning i gruppen. De klarade sig väldigt självständigt och de lyckades bli klar med mer än planerat och var ute i god tid med anmälan till fredagens tävling. Vi utförde även en liten intervju med de enskilda personerna i gruppen för att få en personlig bild över hur de ser på stressnivån i gruppen samt deras syn på CD. Detta var även första gången som vi presenterade CD för dem som koncept, då vi valt att inte nämna det tidigare i oron att det kunde vinkla studien. Dagens release kom iväg någon minut sent och vi hade ett smärre missöde vilket gjorde att releasen inte var fullständigt uppdaterad och saker falerade när kunden testade releasen. Lyckligtvis så löste sig allt tillslut genom diskussion med kunden samt en ny release innehållande den senaste versionen av mjukvaran.

4 Resultat

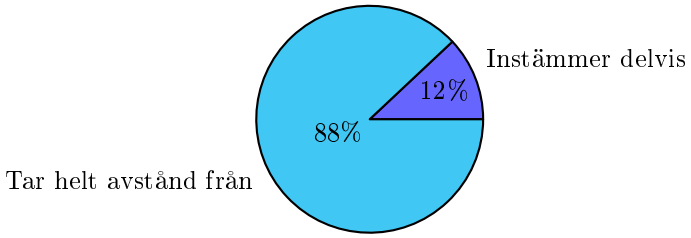
Även med vissa motgångar så har vi lyckats få våra gruppmedlemmar att tillämpa CD. Genom intervjuerna med gruppen såg vi att stressen kring en release var minimal (Figur 2). Vi hade en överlag hög kodvalitet med fullt fungerande releases till kunden. Sista iterationen bröt mönstret, men den stora skillnaden med den releasen jämfört med tidigare är att vårt team inte har kunnat be om vare sig uppskov eller om kunden kunde tänka sig att inte ta med all utlovad funktionalitet i releasen. Eftersom att den sista releasen skulle användas till en tävling i slutet av kursen så krävdes det att viss funktionalitet fanns med. Således fick teamet lämna ifrån sig kod som inte var färdigtestad och inte egentligen hade nått Done Done. Detta ledde till att mjukvaran inte fungerade ordentligt.

Svaren på den allmäna enkäten riktad till övriga studenter på PVG-kursen (Figur 3) visar en bild av delvis stressiga långlabbar men med hög kommunikation och hyfsad effektivitet. Många grupper blev inte klara med sin release i tid och det berodde, över lag, på dålig planering och slarv. Man ser också att alla personer inte klarade av att göra en release utan i vissa fall detta lämnas över till några specifika personer. Slutligen ser man att det för vissa grupper kunde ta upp till två timmar att få till en release. Detta skiljer sig något från det vi har upplevt från vårt team, där alla kunde göra en release, samt att själva releaseprocessen överlag var snabb och problemfri.

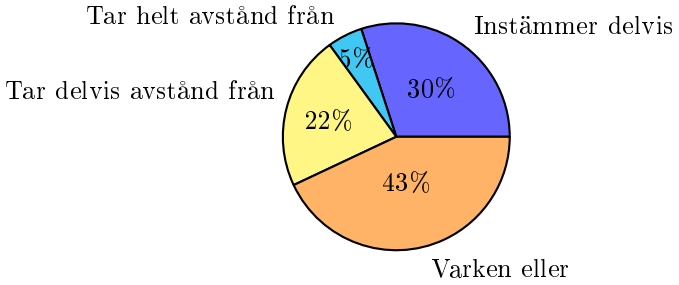
Svaren på enkäten till coacherna (Tabell 1) angående kodkvalitet gav en bild av

att stabiliteten varit generell medel till hög. Datan presenteras nedan. Jämförs detta med vårt teams releaser så framkommer det att vårt team har legat i topp genom alla releaser utom den sista, vilket tyder på att CD kan ha haft en positiv effekt på kodkvaliteten.

När vi sammanställde och analyserade svaren från intervjun med vårt egna team fick vi fram att alla vet hur man gör en release och att de föredrog CD framför att inte ha med det i releaseprocessen. Vi försökte hålla en öppen och levande diskussion och fin en bra inblick i vad de tyckte och tänkte. Gruppen som helhet var väldigt eniga när det kommer till ANTs användbarhet och att de såg CD som något lätt och naturligt. De berättade även om viss stress under de stora releaserna till kunden men skyller det på dålig planering och likaledes dålig estimering.



Figur 2: Resultat gällande stressnivån under en release från den individuella tankestunden.



Figur 3: Resultat gällande stressnivån under en release från den allmänna enkäten.

	1	1b	1c	2	3
Fungerader klockrent	0%	30%	20%	20%	88%
Fungerade bra	60%	60%	10%	40%	12%
Fungerade hjälpligt	40%	0%	0%	30%	0%
Fungerade inte alls	0%	10%	0%	10%	0%
Hade inte release 1c	-	-	70%	-	-

Tabell 1: Tabell över övriga teams kodkvalitet vid release, för respektive release. Releace 1c var speciell och utfördes endast på explicit krav från kunden.

5 Diskussion

Gruppen vi utförde detta experiment på var inte det mest erfarna team när det kommer till programvaruutveckling vilket kan vara orsaken till att vi inte kan se en stor kontrast med andra grupper. Likaså är den här simuleringen av ett projekt väldigt kort, en arbetsvecka drygt, vilket medför att den tid vi spenderat på att lära in våra rutiner inte blev helt återbetalt. Med det i åtanke tror vi fortfarande att vi tjänade på CD vilket är en åsikt vi delar med teamet. De menar att stressmomentet inför en release inte berodde på releasen i sig utan att få med all funktionalitet i tid. ANT-scriptet avlastade teamet mycket från dess att det började nyttjas.

Det vi ser som ett hinder med CD är inlärningskurvan. Denna är nog än värre för en grupp med erfarna utvecklare som är vana att göra det på sitt sätt gentemot den situationen vi haft med oerfarna elever som vill lära sig. Detta iaktogs tidigt under kursen då vi körde en övning under planeringsmöten där vi fick se deras inställning till kursen. Under kursens lopp har vi sett hur majoriteten gått från att vara ivriga att lära sig nya saker mot att mer känna sig som fångar som tvingas gå till dessa lektioner. Anledningen till detta är inte gruppen eller projektet, vilket de vid upprepade tilfällen lovordat. Snarare berodde motviljan att deltaga aktivt på att tentor började närma sig framåt slutet av projektet samt att det började kännas att projektet slukade väldigt mycket tid. Dessutom tyckte de att kursen inte längre hade särskilt mycket mer att lära ut.

Under intervjuerna som hölls så förklarades det för deltagarna vad vi hade gjort för att tillämpa CD med Kanban, testning, DoneDone och ANT och hur vi anpassat detta just för CD gentemot de andra grupperna. Teamet ansåg dock att alla dessa delarna var simpla och självklara för dom och ser inte hur något skulle bli bättre om man strykt en del. Storleken på vår DoneDone bidrog även till att det näst intill krävdes att läsa den på nytt var gång då den var svår att lägga på minnet. Detta gjorde även att den följdes.

Det är alltid lätt att vara efterklok och frågan är då vad hade man gjort om man fått möjlighet att göra om denna studie. Troligtvis hade vi inte ändrat allt för mycket i själva strukturen och experimentet då vi såg goda trender och att få köra det på en till grupp hade mer data för en djupare analys. Däremot hade vi nog ändrat vårt bemötande till gruppen. Vi var för mycket coacher i stunder som de hade behövt en mentor, däremot är detta väldigt spekulativt då det inte

är säkert man hade hamnat med en lika dan grupp och var grupp är unik.

Vi är osäkra på om vi vill rekommendera CD i studentprojekt av denna typ. Projektet är för kort för att till fullo skörda frukterna av de fördelar CD kan presentera. Dock upplevde vi de tidigare nämna fördelarna att teamet fick struktur på en release, kände trygghet i det fungerande ANT-scriptet och att releases innehöll fungerande, stabil kod. Vi fick dessutom kvitto på stabiliteten med resultatet från enkäten till coacherna (Tabell 1). Vi skulle gärna rekommendera CD i längre projekt, och förespråkarna, såsom J. Humble, menar att det fungerar utmärkt i arbetslivet.

6 Tack

Slutligen skulle vi vilja tacka vårt team, Team08 2013, för den enormt roliga upplevelsen att få vara coacher och för alla trevliga fikapauser och fantastisk god allergianpassad fika. Vi skulle även vilja ägna ett tack till Tommy Kvant och Fredrik Åkerberg för hjälp med att bolla idéer och tips.

Referenser

[1] J. Humble and D. Farley. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley, 2010.

[2] Kent Beck, Embracing Change with Extreme Programming, IEEE Computer, Oct 1999

[3] Jez Humble, Chris Read, and Dan North. The Deployment Production Line. In Proceed-ings of the conference on AGILE 2006, AGILE '06, pages 113–118, Washington, DC, USA, 2006. IEEE Computer Society. `http://continuousdelivery.com/wp-content/uploads/2011/04/deployment_production_line.pdf`, besökt 2013-02-18.

[4] Jez Humble, Joanne Molesky, Why Enterprises Must Adopt Devops to Enable Continuous Delivery, Cutter IT Journal Vol. 24 No. 8, August 2011

[5] Jez Humble, Agile Release Management, ThoughtWorks, 2010

[6] Mary Poppendieck and Tom Poppendieck, Implementing Lean Software Development. Addison-Wesley, 2006

[7] InfoQ, Kanban Applied to Software Development: from Agile to Lean, `http://www.infoq.com/articles/hiranabe-lean-agile-kanban`, senast kontrollerad 5 mars 2013

[8] N. Fors och N. Hansson, Kanban i Extreme Programming, Department of computer sience 2010, LTH

[9] The Apache Ant projekt, `http://ant.apache.org/`, senast kontrollerad 5 mars 2013

[10] Praqma, `http://www.praqma.com/`, senast kontrollerad 5 mars 2013

Appendix A: Den allmäna enkäten

Nedan presenteras den enkät som skickades ut till alla grupper.

1. Vilken grupp tillhör du?
2. Är planeringsmötena stressiga?
    - Ja
    - Ibland
    - Nej
3. Vad tycker du om långlabbarna?
    (a) Stressiga
        - Instämmer helt
        - Instämmer delvis
        - Varken eller
        - Tar delvis avstånd från
        - Tar helt avstånd från

    (b) Effektiva
        - Instämmer helt
        - Instämmer delvis
        - Varken eller
        - Tar delvis avstånd från
        - Tar helt avstånd från

    (c) Bra kommunikation
        - Instämmer helt
        - Instämmer delvis
        - Varken eller
        - Tar delvis avstånd från
        - Tar helt avstånd från

    (d) Högt sammarbete
        - Instämmer helt
        - Instämmer delvis
        - Varken eller
        - Tar delvis avstånd från
        - Tar helt avstånd från

    (e) Många parbyten
        - Instämmer helt
        - Instämmer delvis
        - Varken eller
        - Tar delvis avstånd från
        - Tar helt avstånd från

4. Hur fungerar en release för ert team?
5. Känner du att du skulle klara av att göra en release?
6. Blir era releaser klara i tid?
    - Ja
    - Nej
7. Om nej på föregående fråga, varför inte?
8. Finns det något verktyg som hjälper vid en release?

Appendix B: Individuell tankestund

När diskusionen hölls med gruppmedlemmarna så var detta de frågor som diskussionen hade i fokus.

- Vi har sett att era releaser har vart stressiga, tycker du det och vad tror du det beror på ?
- Genom att vi som coacher lade tidig fokus på ANT hur påverkade det arbetet?
  - Hjälpte det?
  - Stjälpte det?
- Tror ni att det hade sett annorlunda ut och vi kommit mycket längre/kortare om vi aldrig jobbat efter CD?
- Är det någon del av CD som varit jobbig?

Appendix C: Releaser i andra team

Den här enkäten skickades ut till alla coacher efter sista releasen.

1. Release 1
   Hur väl fungerade releasen?
   - Fungerade klockrent! Kunden hade inga klagomål
   - Fungerade bra, kunden hade bara lite ytterligare att önska
   - Fungerade hjälpligt, kunden var inte helt nöjd
   - Fungerade inte alls, programmet kraschade/startade ej
2. Release 1b
   Hur väl fungerade releasen?
   - Fungerade klockrent! Kunden hade inga klagomål
   - Fungerade bra, kunden hade bara lite ytterligare att önska
   - Fungerade hjälpligt, kunden var inte helt nöjd
   - Fungerade inte alls, programmet kraschade/startade ej
3. Release 1c
   Hur väl fungerade releasen?
   - Fungerade klockrent! Kunden hade inga klagomål
   - Fungerade bra, kunden hade bara lite ytterligare att önska
   - Fungerade hjälpligt, kunden var inte helt nöjd
   - Fungerade inte alls, programmet kraschade/startade ej
   - Vi hade inte release 1c
4. Release 2
   Hur väl fungerade releasen?
   - Fungerade klockrent! Kunden hade inga klagomål
   - Fungerade bra, kunden hade bara lite ytterligare att önska
   - Fungerade hjälpligt, kunden var inte helt nöjd
   - Fungerade inte alls, programmet kraschade/startade ej
5. Release 3
   Hur väl fungerade releasen?
   - Fungerade klockrent! Kunden hade inga klagomål
   - Fungerade bra, kunden hade bara lite ytterligare att önska
   - Fungerade hjälpligt, kunden var inte helt nöjd
   - Fungerade inte alls, programmet kraschade/startade ej
6. Kommentarer