Peter Lindberg
Computer Programmer, Oops AB
mailto:peter@oops.se
http://oops.se/

28 februari 2002

Planeringsspelets mysterier, del 1

Om jag ska spela ett sällskapsspel för första gången så vill jag att reglerna ska vara så enkla som möjligt. Jag brukar skumma igenom regelhäftet fort och skulle tappa tålamodet fort om det var för tjockt. För något år sedan fick vi ett Master Mind-spel. Jag skummade igenom reglerna och under de första partierna vi spelade så tyckte jag att spelet var superlätt. Efter ett tag läste jag dem igen och insåg att man inte skulle sätta de små "piggarna" (som visar om motståndaren prickat rätt) på motsvarande position som de lite större färgade piggarna. Efter den upptäckten blev spelet mycket svårare (och roligare).

När det gäller hur planering sker i extremprogrammering (XP) så förklaras det i litteraturen som om det vore ett sällskapsspel och det är också därför det kallas planeringsspel (eng. "planning game"). I motsats till min erfarenhet med Master Mind-spelet så kan planeringsspelet vara svårspelat när man inte har förstått reglerna. Min erfarenhet är att dessa regler många gånger är för komplexa för nybörjarspelaren. Dels ska man vänja sig vid att nedteckna kraven på registerkort bara med några få meningar, dels ska man lära sig hur man formulerar dessa meningar och hur mycket varje kort bör omfatta. Det finns många olika speldrag att välja bland och att förstå när varje drag är kan göras och varför kan vara svårt.

Jag ska i ett par artiklar försöka förklara planeringsspelet på ett sätt som jag tror kan vara lättare. I sista artikeln kommer jag dock att upprepa spelreglerna såsom de beskrivs i litteraturen. Detta eftersom jag är övertygad om att de är till nytta när man väl har förstått principerna. Dessa artiklar är inte menade som en heltäckande introduktion till varesig XP eller planeringsspelet, utan jag förutsätter att du som läser detta är bekant med XP sedan tidigare. I denna första artikel kommer jag först att ytligt beskriva planeringsspelet och dela upp det i tre olika delar, samt gå in på vanliga frågor och problem gällande den första av delarna.

Planeringsspelets kärna

All planering handlar om att fastställa vad som ska göras, hur lång tid det tar, i vilken ordning det ska göras samt vem som ska göra vad. Omfattning. Tid. Prioritering. Tilldelning. I XP vet man redan hur lång tid det tar eftersom man kör med fasta iterationer (aldrig någonsin ruckar man på ett iterationsslut). Istället intresserar man sig av hur mycket man hinner med på denna tid (2-3 veckor). I XP handlar planering alltså om tre saker: omfattning, prioritering och tilldelning.

Om vi börjar med omfattning så behöver man ta reda på vad man härnäst vill ha ut av systemet som byggs, vilka förändringar av och tillägg till det som önskas. Dessa "krav" nedtecknas inte i detalj, utan i form av en användarberättelse (eng. "user

story”) på ett par meningar på ett registerkort, för varje planerad förändring eller tillägg.

Under diskussion identifieras ett antal användarberättelser. “Storleken” på dessa behöver sedan uppskattas, av programmerarna, för att man ska veta att man har tillräckligt med dem för att fylla en hel iteration. Har man inte tillräckligt med kort får man hitta på mer att göra. Har man för många så har man större möjlighet till att prioritera effektivt, i synnerhet om varje berättelse är i lagom storlek.

Efter omfattning kommer prioritering. Du som kund måste fördela de kort som har skrivits i tre högar framför sig på bordet – en hög för prioritet 1, en för prioritet 2 och en för prioritet 3. Vitsen med att ordna berättelserna på detta sätt är att man, om man inte skulle hinna med alla under iterationen, åtminstone har hunnit med de allra viktigaste.

När du som kund får höra vad programmerarna satt för storlekar på dina berättelser så kan du komma att prioritera berättelserna annorlunda än du först tänkte. En större storlek innebär längre tid att implementera och alltså ett högre pris för den önskade förändringen. En högre pris kan innebära att du vill prioritera ned en tidigare högt prioriterad förändring, medan en lägre prioriterad blir mer intressant om du får reda på att priset är väldigt lågt.

Till slut har vi tilldelning, vilket i XP inte går till så att någon bestämmer vem som ska göra vad, utan istället anmäler sig varje programmerare för att implementera de användarberättelser han eller hon är intresserad av. Man kan uttrycka det som att de tilldelar sig själva uppgifter. Samma sak, fast omvänt och mycket trevligare!

Jag ska nu försöka reda ut vanliga problem inom vart och ett av områdena omfattning, prioritering och tilldelning. Är du insatt i XP och misstänksam till naturen kanske du tycker att det finns luckor i beskrivningen hittills. Var lugn, jag kommer till dem.

Omfattning

De vanligaste problemen när det gäller omfattning handlar om hur man skriver användarberättelser och hur man uppskattar deras storlek. Detta problem tror jag beror på att man har en känsla av att allt måste finnas på papper för att “räknas”. I XP är det väldigt mycket som inte finns på papper och därför finns det mycket att oroa sig för tills dess att man inser det överlägsna i muntlig kommunikation.

Problem med storleksuppskattning har att göra med att man fortfarande ser det som tidsuppskattning. Jag undviker avsiktligen detta ord av denna anledning. Tid är något mycket abstrakt och alla försök att fånga den är dömda att misslyckas. XP:s lösning på detta problem är dock briljant, vilket jag hoppas kunna visa här.

Hur skriver man en användarberättelse?

Det finns många olika förvirrande bud när det gäller att skriva användarberättelser. Man får höra att man ska skriva så kortfattat som möjligt och att det inte ska vara en teknisk beskrivning av en implementation, utan en enkel berättelse om en funktion såsom användaren kommer uppleva den. Det sägs att varje berättelse måste handla

om affärsnytta och när det faktiskt finns behov av förändringar som inte står i direkt relation till affärsnytta så ska man ändå försöka finna någon affärsnytta med den, annars har den inget berättigande.

Min åsikt är att man kan skriva precis vad man vill på korten, sålänge som alla i teamet utifrån vad som står på dem har lätt att minnas diskussionen som fördes då kortet skrevs. Det brukar sägas att korten är "löften om vidare diskussion", dvs att de inte är till för att dokumentera exakt alla krav som ska uppfyllas, utan att man för varje kort kommer ha flera diskussioner med olika inblandade personer. Det är minst lika viktigt att betrakta dem som "påminnelser om tidigare diskussion", att korten ska hjälpa en att minnas vad som diskuterats.

Det kan underlätta att betrakta korten som "punkter på en attgöralista". Tänk dig att du skriver en sådan lista. Det du skriver vid varje punkt är oftast en kort och kärnfull fras, i stil med "Skaffa nya vinterdäck", "Köp 3 l mjölk" eller "Tvättstugan mån 19-22". Du tvekar knappast över formuleringen eller vad som behöver nämnas: "Fattar jag att det är kläder jag ska tvätta på måndag kväll?" eller "Måste jag skriva att jag ska köpa mjölken på Vivo på Upplandsgatan?" Vid planeringen är samtliga närvarande och frågor kommer ställas naturligt under diskussionen. Alla frågor och svar behöver inte nedtecknas eftersom sannolikheten att ingen skulle minnas dem är mycket liten.

Hur mycket eller hur lite du skriver spelar ingen roll. Känner du att du behöver tio meningar på dig är det helt okej. Det skulle definitivt medföra att det bättre skulle hjälpa minnet, eftersom det är mer text och således borde innehålla fler nyckelord från diskussionen. Någonstans finns det säkert en gräns för hur mycket man bör skriva. XP handlar om att exploatera kommunikation och man vill undvika att den som tar emot kortet upplever det som en "färdig specifikation" och därför låter bli att ställa frågor. Risken för att detta ska inträffa är dock begränsad, eftersom det inte ryms hur mycket text som helst på ett kort.

Skulle du vara tveksam över formuleringen så behöver du bara läsa upp vad du har skrivit och fråga de andra vad de tycker. Kom ihåg att det är diskussionen och samarbetet som är viktigast.

Hur "stor" ska en användarberättelse vara?

En annan sak som vållar problem, men som du som kund i själva verket inte behöver tänka på, är hur omfattande en användarberättelse ska vara, dvs hur stor portion funktionalitet som ska beskrivas på varje kort. Detta är enbart ett problem om man som kund på egen hand försöker skriva användarberättelser. Då är det omöjligt att avgöra om det man skriver är något som tar flera veckor eller något som tar ett par dagar. Vänta istället tills mötet med programmerarna och låt *dem* säga när en berättelse är "för stor" eller "för liten".

Vem ska skriva användarberättelserna och när ska de skrivas?

XP-litteraturen är noga med att påpeka att det är kunden som ska skriva användarberättelserna. Detta betyder inte att han eller hon måste skriva dem före mötet med programmerarna. Tvärtom är det bättre att vänta med att skriva dem tills under mötet. Anledningen till detta är att man aldrig kan veta vilka upptäckter man gör

under diskussionen kring dem. Dessutom tror jag det bidrar till att alla närvarande bättre förstår de användarberättelser som planeras om de formuleras och skrivs ned tillsammans.

Det viktiga är samarbetet och diskussionen. Jag tycker mig ha anat en otålighet hos några över att alla medlemmar i teamet sitter samlade och diskuterar, men om man tänker på det så finns det inget effektivare sätt. Låt säga att planeringsmötet hålls mellan två par ögon. För att förmedla detta till teamet behöver man antingen skriva ned det, vilket medför stor risk för luckor samtidigt som det tar längre tid både att skriva och att tillgodogöra sig informationen genom att läsa den. Ska man förmedla det muntligt – varför inte göra det vid ett och samma tillfälle? Är alla samlade ökas chansen att viktiga frågor ställs.

Poängen med att det är kunden som skriver ned berättelserna är, såvitt jag kan se, att betona att de "ägs" av kunden och att säkerställa att han eller hon verkligen förstår innebörden av det som "beställs". Eftersom jag själv inte tycker att formuleringen är kritisk så tycker jag också att det viktiga inte är vem som skriver utan att samtliga närvarande kopplar samman vad som skrivs med vad som diskuteras.

Ju vanare du som kund blir med XP-processen, desto mer kan du förbereda före mötet med programmerarna. Ingenting av det du nedtecknar – varesig det är på registerkort eller en enkel lista med punkter – är dock att betraktas som "spikat". Oavsett hur varm i XP-kläderna du är så kommer det fortfarande kunna upptäckas nya infallsvinklar under diskussionen.

Ha gärna en uppfattning om vilka användarberättelser du vill ska skrivas under planeringsmötet, men vänta med att skriva dem tills under mötet, om inte annat så för att underlätta för de andra på mötet att förstå och minnas diskussionen. Känner du dig trygg med att själv skriva och formulera korten så gör det. Behöver du hjälp så låt det ske i samarbete med de andra. Huvudsaken är att främst du själv känner dig säker på innebörden i den diskussion som varje kort representerar, men även att de andra på mötet gör det.

Hur många användarberättelser hinner vi med?

För att avgöra hur många användarberättelser man hinner med så måste man ha en uppfattning om storleken på varje berättelse. Jag skriver "storleken" eftersom jag menar att man i XP försöker undvika att uppskatta tid. Det som kallas "tidsuppskattning" är med rätta många programmerares mardröm. Om det vore så att man som programmerare var en helt isolerad organism, skyddad mot alla former av yttre påverkan – och dessutom var så briljant, att man från vad som oundvikligen är bristfällig information, exakt visste vad som ska göras – så vore tidsuppskattning möjlig.

I XP uppskattar man alltså storleken och inte tid, även om det inte beskrivs exakt så i litteraturen. Det talas istället om "idealtid", vilket är den tid den isolerade organismen jag beskrev skulle behöva för att göra något. Min erfarenhet är att det blir förvirrande att använda "idealveckor" eller "idealdagar" som mått samtidigt som man talar om verklig tid. Det har föreslagits (bl a i boken *Extreme Programming Installed* av Ron Jeffries, m fl) att man av denna anledning ska använda andra ord för detta – t ex

“punkter”. Låt oss ta reda på om vi kan klara oss utan begreppet idealtid helt och hållet.

Storleken på varje användarberättelse uppskattas kollektivt av programmerarna som närvarar vid mötet. Något som många tycker underlättar är om man ställer berättelsen i relation till andra som redan har fått sina storlekar uppskattade – eller ännu hellre till sådana som redan har implementerats. Det är ofta lätt att säga om en berättelse grovt uttryckt är större, mindre eller jämförbar med en annan. Om man jämför den med en som uppskattats till en punkt och tycker att den är större – men att den är mindre i jämförelse med en som uppskattats till två punkter – så sätter man dess storlek till 1,5 punkter. För enkelhetens skull och för att man inte behöver större noggrannhet, så avrundar man till hela eller halva punkter.

Vid planeringen av en iteration bryr man sig bara om hur mycket man hann med under den föregående iterationen. Om man hann med 3,5 punkter så planerar man att hinna med 3,5 punkter även denna omgång – oavsett hur högt man siktade förra iterationen.

Frågan är då vad man ska titta tillbaka på när man planerar första iterationen. I början av min bana som professionell programmerare så hörde jag ett skämt om att man behövde multiplicera vars och ens tidsuppskattningar med π (pi) för att få reda på hur lång tid det skulle ta i verkligheten. Eftersom vi redan vet den verkliga tidsåtgången, alltså iterationslängden, så måste vi dela med π för att få reda på hur mycket vi kan göra under en iteration. För att göra det enkelt så kastar vi de jobbiga decimalerna i π – vårt magiska tal är 3. Denna formel ger då, att med en iterationslängd på tre veckor, så är varje programmerare kapabel att implementera motsvarande en punkt den första iterationen (3 / 3 = 1). Är iterationslängden två veckor blir svaret 0,5 punkter efter avrundning (2 / 3 = ca 0,67). Med fyra programmerare hinner man med fyra punkter på tre veckor och två punkter på två veckor.

Har vi då klarat oss utan att befatta oss med idealtid? Kanske! När ni som programmerare vid planeringen av er första iteration ska uppskatta storleken på den första användarberättelsen, så tänker var och en av er: “Jag har en punkt ‘att spendera’ denna iteration som varar tre veckor. Hur många användarberättelser i den här storleksordningen skulle jag hinna med att implementera?” Blir svaret två är storleken på berättelsen 0,5 punkter. Blir svaret att man inte skulle hinna med den så är den för stor och måste brytas ned i mindre. Blir svaret fler än två bör man försöka slå samman den med någon eller några andra användarberättelser. Hur som helst, efter att bara ha uppskattat storleken på en berättelse har man genast något att jämföra med vid uppskattningen av de övriga.

Genom att vända på steken på detta sätt kommer vi förstås inte helt undan idealtidsbegreppet. Det finns ju någonstans därunder de antaganden man gör. Men några kanske tycker att detta är ett mer lättfattligt sätt att angripa problemet på. Föredrar man det andra sättet med idealtid och hastighet (eng. “velocity”) så kanske denna övning har hjälpt till att förtydliga.

Vad gör vi om vi “vet” att vi hinner med mer än förra iteration?

En iteration jag var med om hade vi bara planerat med en enda stor berättelse för alla programmerare. Jag minns inte exakt vad det handlade om, men jag minns att vi inte hann klar med den. Regeln att man planerar att hinna med så mycket denna iteration som man hann med under förra, skulle i vårt fall ha betytt att vi inte kunde räkna med att hinna med någonting alls. Vi tänjde lite på reglerna och planerade in dels att slutföra det vi inte hann med och dels att genomföra lite nya saker. Dessutom la vi på minnet att försöka låta bli att skriva så stora användarberättelser i fortsättningen.

Ett vanligare scenario är att man har en känsla av att man hinner med mer denna gång än förra. Det kan vara av revanschlusta, eller så handlar det bara om att någon som varit på semester eller varit sjuk kommit tillbaka till arbetet. Det är dock klokt att hålla fast vid denna regel ändå, men har man den känslan kan man passa på att skriva några extra berättelser, sätta storlekar på dessa samt prioritera dem, så att man inte behöver kalla samman till ett möte om det verkligen blir som man tror.

Nästa artikel

Jag har behandlat några problem och frågor jag dels har stött på själv, men även sett återkomma på mailinglistor och i andra fora. Just nu känner jag mig klar med området Omfattning och i nästa artikel kommer jag fortsätta med nästa område: Prioritering. Den artikeln är dock ännu inte skriven, så om du har något du vill bidra med så kan jag kanske ta dem med den eller kommande artiklar. Jag vore tacksam för synpunkter på det jag har skrivit, samt korrigering av språkliga fel.

Diskutera denna artikel på det svenska XP-communityts egen Wiki.

http://oops.se/cgi-bin/wiki?ExtremProgrammering