Översikt över vanliga metoder och verktyg

Albin Engström

Sammanfattning

I rapporten så ser vi att det finns många metoder och verktyg att använda vid mjukvaruutveckling. Flera som på olika sätt försöker hantera samma problem eller underlätta samma sak.

Särskilt metoderna för hur man strukturerar arbetet är många och varierade och fungerar på mycket olika sätt.

Innehållsförtecking

Inledning

I denna rapport så ska jag gå igenom och jämföra en rad olika metoder och verktyg för att underlätta och förbättra mjukvarutveckling. Det finns mycket a prata om inom detta ämne men jag tar bara upp ett urval av de vanligaste inom sina respektive områden.

Metoder för utveckling av mjukvara

Kort översikt

The Waterfall Model

Refereras I texten som ”TWM”.

Det utmärkande dragen hos TWM är att utvecklingen sker i ett antal klart definierade steg som följs i en förutbestämd nedstigande ordning, som ett vattenfall.
Vad som ska göras I dessa steg försöker man etablera så specifikt som möjligt innan man börjar.

De stegen är i originalversionen av modellen:

1. Requirements Specification
2. Design
3. Construction(Coding)
4. Integration
5. Testing and Debugging
6. Installation
7. Maintenance

Vad stegen är och när de görs kan variera I de olika versionerna av modellen. Modellen ovan är den första versionen och även personen som skapade modellen, Winston W. Royce, har skapat nya versioner.

Extreme Programming

Refereras i texten och även allmänt som ”XP”.

XP tillhör kategorin a*gila metoder,* och precis som andra så utvecklas programvaran i många korta iterationer där målet är att alltid ha ett fungerande och buggfritt program efter någon funktion lagts till i koden.
Dock lägger XP extra mycket tid och fokus på testning, med filosofin att om lite testning leder till att några problem upptäcks så borde mycket testning uppdaga många problem.
Denna testning görs både direkt på ny kod, enhetstestning, och med kunden, acceptanstestning.
Detta för att se om koden fungerar som den ska och som kunden vill.

Scrum

En agil metod.
Scrum arbetar runt principen att kunden ofta ändrar sig, lägger till och tar bort funktioner som de vill ha och att man därför inte kan planera arbetet in i minsta detalj för långt i förväg.
Istället så fokuserar man på att snabbt kunna anpassa sig efter vad som ska göras.
Detta görs bland annat genom att teamet ofta har möten så att de alltid har koll på vad som ska göras och vad andra gör.

Dynamic Systems Development Method

Refereras I texten och allmänt som DSDM.

Den nyaste versionen av DSDM är den som behandlas här, den kalls för Atern.

Precis som Scrum och XP så definieras DSDM som en agil metod.
Men den lägger stor fokus på att undvika vanliga problem vid mjukvaruutveckling, bland annat missade deadlines och överskridna budgetar.
Därför fastställs de noga vid start och olika metoder implementeras för att hålla sig till dem.

Jämförelse

Komplexitet

TWM är i sig självt den ”enklaste” metoden. Den har en klart definierad uppsättning steg vars ordning är fastfälld.

Komplexiteten kommer från själva projektet, inte metoden.

XP och Scrum är mer komplexa, XP har flera listor med regler, principer och värderingar som man ska tänka på och använda i projektet. Scrum har inte lika många saker att tänka på men har istället en uppsättning regler om hur möten ska genomföras och vilka sorters möten som ska ske när.

DSDM har 8 stycken principer som man ska följa, dock har de principerna flera andra saker på undernivåer. Vilket gör den aningen mer komplicerad än TWM.

Skala

Scrum är tänkt för team på 3-9 personer.

TWM kan appliceras på team av alla möjliga storlekar, det är själva arbetet som avgör om det är för få eller för många inblandade.

DSDM har samma förutsättningar som TWM, dock har den ett fokus på kollaboration med ett team.

XP brukar sägas kunna användas för team på 12 eller mindre personer.

De olika metoderna passar olika bra med olika storlekar på team, men möjligheten att dela upp ett projekt i flera delar med fler team finns alltid. Dock så är det olika lämpligt med olika metoder.

Metoder för testning av programvara

Kort översikt

Eclipses inbyggda debug system

Detta system används främst för enhetstestning och integrationstestning under utvecklingens gång.

Man kan köra programmet och om något fel inträffar så får man information om vad felet är, var det inträffar, vilken klass och metod som är ansvarig och vilka värden de inblandade variablerna har.

Man kan även “frysa” programmet mitt under exekvering för till exempel kunna undersöka variablers värden innan något problem faktiskt har uppstått vilket är användbart om orsaken till problemet ligger högre upp i en händelsekedja än där buggen först märks.

Som exempel har jag den här funktionen:

```
private static Socket getSocket(PlayerSetup playerSetup)
                  throws FoxgameNoGameToParticipateException {
            try {
                  InetAddress address = InetAddress
                              .getByName(playerSetup.serverIpAddress);
                  return new Socket(address, playerSetup.gamePort);
            } catch (IOException e) {
                  throw new FoxgameNoGameToParticipateException(playerSetup);
            }
      }
```

Det är en del av en klass som hanterar kommunikationen mellan ett rävspel och dess spelare. Den kastar en exception eftersom det inte finns något spel för tillfället.

Man kan då bland annat se information om playerSetup objektet.

Codeblocks exekvering benchmark

En väldigt simpel version of testing är att ta tid på hur lång tid programmet eller en del av programmet tar att göra sitt jobb.

Det finns många sätt att mäta det från att lägga in mätning i koden till dedikerade program.

Det jag har testat är Codeblocks inbyggda system. När man exekverar ett program så står det hur lång tid det tog.

För mera avancerade mätningar kan man bland annat mäta hur mycket ram programmet använder och hur mycket hårddisk som utnyttjas.

Alla saker som man har intresse av att minimera I ett program är användbart att testa och förhoppningsvis göra ändringar för att förbättra.

Denna typ av testning kan ingå i enhetstestning, integrationstestning, systemtestning och acceptanstestning.

Jämförelse

Att jämföra två så olika saker är lite svårt, men jag har kommit på några lämpliga områden.

Användarvänlighet

Hur enkla de är att använda skiljer sig mycket. Att förstå sig på och navigera Eclipses debug fönster kräver en del medans Codeblocks helt enkelt ger en siffra.

Att sedan använda informationen man får kan variera stort I hur komplicerad det är i båda fallen.

Att lokalisera och åtgärda en bugg kan vara mycket simpelt men det kan också vara väldigt svårt.

Att optimera ett program är sällan riktigt lika enkel men kan bli like svårt beroende på i vilket område man vill förbättra.

Användbarhet

Att använda ett system för att hitta och åtgärda buggar är något som alltid är användbart. Det är sällan något mer komplicerat en de simplaste program skrivs utan några som helst problem.

Optimering däremot är i många fall inte lika viktigt. Många program är inte så krävande att de nuförtiden behöver oroa sig för att dators inte ska orka.

I vissa fall är det dock mycket viktigt om programmen till exempel är mycket krävan eller ska använda på stor skala.

Stödjande verktyg

Översikt

Git

Git är ett system för versionshantering med fokus på samarbete mellan personer som inte är på samma nätverk.

Alla som arbetar på ett projekt har en egen fullvärdig kopia vilket bidrar till att risken att förlora all data är liten.

Den har ett bra stöd för att hantera att många gör olika ändringar samtidigt och hjälper till med att hålla koll på vem som ändrat vad och med att får ihop all ändringar till samma projekt. Eller att dela upp projektet i olika grenar om de deltagande vill göra saker på olika sätt.

Github

Github är ett extra lager av funktionalitet byggt ovanpå Git. De mest noterbara tilläggen är att den ger ett grafiskt webbaserat gränssnitt till Git och tillhandahåller en extern server där en kopia av projektet finns. Vilket bland annat är praktiskt när man enasm arbetar med ett projekt eftersom det då blir en backup.

Andra tillägg är ett system för att hantera buggrapporter och önskemål från användare.

Apache Subversion

Subversion är också ett system för versionshantering och använder ett klient-server system där klienterna synkroniserar mot en server.

Ändringar som ska skickas till servern är atomiska vilket betyder att de antingen genomförs helt eller inte alls. På det viset undviks problem om processen avbryts innan den är klar. Till exempel om nätverket krånglar.

Jämförelse

Att jämföra så olika saker är även här lite svårt, men jag kan använda samma områden här med.

Användarvänlighet

Git och Subversion är båda helt hanterade genom en terminal. Även om personliga preferenser varierar så kan man nog säga att ett grafisk användargränssnitt är mer användarvänligt för de flesta. Något som Github bland annat ger Git.

Subversion har också ett antal GUI.

Användbarhet

Alla här faller inom samma användningsområde så alla är i stort sett lika oavvändbara. Framförallt i syfte av att hantera stora projekt med flera personer men även för att ha säkerhetskopior av arbetet och mycket annat.

Dokumentationsverktyg

Översikt

Doxygen

Doxygen är en generator för dokumentation som hämtar sin information från kommentarer i koden. Detta gör att den är lätt att använda eftersom man redan borde ha kommentarer i koden, det ända man behöver göra är att använda en korrekt syntax.

Den har också koll på vad själva koden är så man kan lätt undersöka hur koden ser ut till dokumentationen man läser.

Github

Github har ett system för att göra en wiki för ett projekt. Där kan man manuellt göra alla typer av dokumentation. Bland annat förklaringar för vad programmet är och guider till hur man använder det.

Jämförelse

Användarvänlighet

Man kan säga att eftersom Github har ett GUI som man kan jobba med så är det mer användarvänligt. Dock så görs det mesta av arbetet för Doxygen i koden av kodaren och den personen eller personerna har då troligen inga problem med att skriva kommentarer i ett specifikt syntax, man behöver dock lära sig den.

Flexibilitet

Githubs wiki system är mycket mer flexibelt eftersom man kan använda den för all möjlig dokumentation rörande programmet. Doxygen är fullt fokuserad på att dokumentera själva koden.

Källförteckning

https://en.wikipedia.org/wiki/GitHub

https://en.wikipedia.org/wiki/Git_%28software%29

https://en.wikipedia.org/wiki/Doxygen

https://en.wikipedia.org/wiki/Software_documentation#User_documentation

http://subversion.apache.org/

https://en.wikipedia.org/wiki/Apache_Subversion

https://en.wikipedia.org/wiki/Version_control#Distributed_revision_control

https://en.wikipedia.org/wiki/List_of_version_control_software

https://en.wikipedia.org/wiki/Dynamic_systems_development_method

https://en.wikipedia.org/wiki/Extreme_programming

https://en.wikipedia.org/wiki/Scrum_%28software_development%29

https://en.wikipedia.org/wiki/Waterfall_model

https://en.wikipedia.org/wiki/Software_development