Beteckning:_____

HÖGSKOLAN
I GÄVLE

Akademin för teknik och miljö

Team Foundation Server 2010 – En utvärdering av mätetal för projektkvalitet

Dennis Tapper & Edvin Andersmats
Juni 2011

Examensarbete, 15 högskolepoäng, C
Datavetenskap

Datavetenskapliga programmet
Handledare: Göran Sundberg
Examinator: Fredrik Bökman

Team Foundation Server 2010 – En utvärdering av mätetal för projektkvalitet

av

Dennis Tapper & Edvin Andersmats

Akademin för teknik och miljö
Högskolan i Gävle

S-801 76 Gävle, Sweden

Email:
denka07@gmail.com
eandersmats@gmail.com

Abstrakt

På Trafikverkets centrala funktion IT används Team Foundation Server 2010 för projekthantering. Utifrån detta system det är det möjligt att få fram mätetal som visar information om olika projekt. I denna fallstudie undersöks, med hjälp av intervjuer, vilka av dessa mätetal som kan användas för att visa en relevant bild av kvaliteten på ett systemutvecklingsprojekt. En modell presenteras för hur olika mätetal skulle kunna vägas samman för att fungera som en siffra på ett projekts kvalitet/hälsa. Till sist utvärderas om mätetal från TFS är tillräckliga för att ge en bra bild av kvaliteten på ett systemutvecklings-projekt. Studien kommer fram till att en sammanvägning är möjlig, men kan vara omfattande. Den visar även att mätetal från TFS är tillräckliga för att ge en övergripande bild av ett projekts hälsa.

Nyckelord

Team Foundation Server 2010, mätetal, projektkvalitet, The Iron Triangle.

Innehåll

1 Introduktion

1.1 Bakgrund

Projektgrupper på Trafikverkets centrala funktion IT (cfIT) arbetar oftast mot en tidsgräns för när ett IT-system ska vara klart. När arbetet börjar närma sig slutet kan det i många projekt tillkomma nya krav, vilket i sin tur kan leda till att det blir svårt att hinna klart med allt inom utsatt tid med den arbetskraft som finns tillgänglig. Detta kan även medföra att nya buggar uppstår i ett sent skede, vilket i sin tur kan göra att systemet inte får den kvalitet som det skulle kunna ha.

CfIT använder sig av ett system som kallas Team Foundation Server 2010 (TFS) [1]. Detta används för all programvaruutveckling, och ger stöd för bland annat källkodshantering, automatiska byggen, ärendehantering, rapporter, projekthantering, testning, kvalitetssäkring och regelstyrning. Man kan från TFS hämta ut olika mätetal som beskriver kvaliteten på de systemutvecklingsprojekt som finns registrerade.

Vilka mätetal är möjliga att använda, och hur kan de användas för att ge rätt bild av kvaliteten på olika projekt? Finns det också möjligheter att väga samman dessa mätetal för att t.ex. projektledare ska kunna få en snabb och övergripande bild på den aktuella kvaliteten för ett projekt?

1.2 Syfte

Syftet med detta examensarbete är att utreda vilka mätetal i TFS som kan vara lämpliga att använda för att ge den bästa möjliga bilden av kvaliteten på ett systemutvecklingsprojekt. Dessa ska sedan kunna fungera som en form av ”varningsflagga” för eventuella problem som kan uppstå i systemutvecklingsprojekt. Mätetalen ska kunna ge en helhetssyn över hur ett sådant projekt ligger till i tidsplanen och hur bra kvaliteten på de system som utvecklas är. Alltså ge en uppfattning om ett projekts status.

Frågeställningar:

- Vilka mätetal från TFS kan användas för att ge en relevant bild av kvaliteten på ett systemutvecklingsprojekt, och hur kan dessa användas till detta?
- Kan man väga samman mätetal från TFS för att få ett övergripande värde som beskriver kvaliteten på ett systemutvecklingsprojekt?
- Är TFS tillräckligt för att ge en bra bild av kvaliteten på ett systemutvecklingsprojekt

2 Teoretisk bakgrund

2.1 Kvalitet på systemutvecklingsprojekt

Termen kvalitet kan i vardaglig användning verka självförklarande. I praktiken finns det dock många olika synpunkter på vad kvalitet egentligen är och hur man kan uppnå detta i systemutvecklingsprojekt [5]. Definitionen av ”ett lyckat projekt” är alltså ganska otydlig. Detta styrks av mycket forskning inom området [3][5][6]. Man menar att projektkvalitet därför fortfarande värderas i form av elementen tid, kostnad och produktkvalitet. En triangel som brukar gå vid namnet ”The project triangle” eller ”The iron triangle” [2][3].

En förändring eller ökad fokus på ett element i projektet påverkar automatiskt ett eller två av de andra. Till exempel, om man:

- Hittar en bugg, men anser att det är viktigt att hinna klart inom tidsramen för projektet. Detta kan medföra att man får ökade kostnader och en sämre kvalitet på produkten.
- Märker att det behövs mer resurser, men istället för att tillsätta fler arbetare låter man bli för att spara pengar. Detta kan ge sämre kvalitet på produkten och/eller göra att man inte hinner klart med projektet i tid.
- Man hittar en ny lösning som kan skapa väldigt bra kvalitet på systemet, men det visar sig att man drar ut på tiden och/eller det kostar en del att tillsätta resurser [2].

Kvalitet på ett systemutvecklingsprojekt blir då att försöka balansera dessa element och försöka optimera var och en av dessa [2]. Denna syn på projektkvalitet är dock inte helt fulländad, menar Atkinson [3]. Han tar upp att det finns många andra faktorer som spelar in, men att dessa tre är viktiga och ändå borde användas.

De flesta projekt är allra högst beroende av att produkten håller en hög kvalitet, menar Ferrand et al. [6]. De säger även att:

To achieve quality, organizations should therefore be concerned, throughout their projects' life cycle, with the extent to which the end products will meet their clients' needs and expectations.

2.2 Produktkvalitet

Eftersom projektets kvalitet är starkt kopplat till produktens kvalitet krävs en definition av vad produktkvalitet är.

Ho-Won et al. [8] menar att en mjukvaras kvalitet bestäms av dess förmåga att tillfredsställa användarna av mjukvaran. Borque et al. [9] är inne på samma spår och beskriver en mjukvaruprodukt som en svart låda vars uppgift är att stödja användarnas affärsprocesser. De menar att detta i sin tur leder till att affärsbehoven blir en drivande kraft i utvecklingen av mjukvara. Standarden ISO/IEC 9126 beskriver sex olika kvalitetsattribut vilka i sin tur är indelade i underliggande kvalitetsattribut [7][8]. De sex kvalitetsattributen är:

- Funktionalitet
- Pålitlighet
- Användbarhet
- Effektivitet
- Underhållbarhet
- Portablilitet

Dromey använder sig av dessa kvalitetsattribut för att beskriva kvalitet på mjukvara, men han har även lagt till kvalitetsattributet återanvändbarhet [7]. Han argumenterar

för att det är en viktig del i mjukvarukvalitet och att det inte innefattas helt i de andra kvalitetsattributen och därmed bör vara ett eget kvalitetsattribut.

Han menar att dessa kvalitetsattribut beror på olika kvalitetsbärande egenskaper hos mjukvaran. Dessa egenskaper beskriver han som starkt kopplade till programkoden genom att ange hur de hänger samman med olika delar i koden.

2.3 Agila utvecklingsmetoder

Vad kan man då göra för att uppnå kvalitet på systemutvecklingsprojekt? Traditionella utvecklingsmetoder säger att man ska fokusera på att fastställa alla krav och göra klart större delen av dokumentationen så tidigt som möjligt för att spara tid och kostnad under utveckling. Detta har i många fall istället visat sig leda till att kunden inte fått vad denne velat ha, kostnader har ökat, eller att tidsgränser överskridits. Att använda ett agilt synsätt (används regelbundet på Trafikverkets centrala funktion IT) är ett numera populärt sätt att ta sig an både snabba förändringar i krav från kund samt ett behov av korta arbetsiterationer som krävs i dagens systemutvecklingsprojekt [4][10].

Ordet agil betyder lättrörlig, lättviktig och vig, och detta är vad agila utvecklingsmetoder står för. Det är värderingar, principer och attityder som delas av ett antal metoder som anses agila [4][11]. Gemensamt har alla dessa, enligt det Agila manifestet [12], att de värdesätter:

- **Individer och interaktioner** framför processer och verktyg.
- **Fungerande programvara** framför omfattande dokumentation.
- **Kundsamarbete** framför kontraktförhandling.
- **Anpassning till förändring** framför att följa en plan.

Karaktärerna hos de olika utvecklingsmetoderna kan variera en aning, men de har stora likheter. De börjar oftast med någon form av högnivåplanering och framtagning av nuvarande krav i det första skedet av projektet. Sedan sker release och dokumentation i det sista skedet. Emellan dessa två ligger ett antal iterationer, och det är dessa som karaktäriserar de agila metoderna. Iterationerna i sig är tidsbundna cykler under vilka funktionella och icke funktionella krav diskuteras fram med kund. Kraven analyseras, sedan designas, utvecklas och testas ett delsystem som ska vara redo för leverans [4].

2.4 Utvärdering av kvalitet på systemutvecklingsprojekt

Woodward et al. [1] konstaterar att om man har möjligheten att mäta framsteg i ett systemutvecklingsprojekt kan både ledningen och utvecklarna avgöra om projektet håller sig inom den uppsatta tidsramen eller inte. Att man i ett så tidigt skede som möjligt kan få reda på om man ligger efter eller före i planeringen ger möjlighet till att hantera tid och resurser på ett mer optimalt sätt, och kan på så vis ge ökad kvalitet på mjukvaran som produkt [13].

2.4.1 Testning och utvärdering av kvalitet

I agila projekt sker testning kontinuerligt. Men hur viktig är testning för kvaliteten i dessa, och går det att använda som en indikation av projektets kvalitet?

Enligt Hazzan et al. [14] ska alla utvecklare i ett agilt projekt skriva tester för den kod de skapar. De säger även att det finns många olika former av tester som kan genomföras. Några användbara är:

- Funktionella tester, som ser till att systemet under normala situationer och förväntad indata gör det som det ska göra.
- Prestandatester, som ser till att systemet är tillräckligt snabbt.
- Stresstester, som ser till att systemet kan hantera mer än förväntad mängd data.
- Utforskande tester, som försöker avgöra vad systemet gör när det används på vissa sätt.

De säger också att produktkvalitet och testning är tätt kopplat till varandra, vilket de motiverar genom att säga:

In a traditional project, everyone is responsible for quality, but in agile projects, everyone actually writes tests [14].

Tack vare att testning och produktkvalitet har en mycket stark koppling är det även, enligt Deissenboeck et al. [16], möjligt att utvärdera kvaliteten med hjälp av resultat man får från tester.

2.5 Automatiska verktyg

Enligt Puleio [15] kan och ska utvärdering av kvalitet om möjligt göras med hjälp av automatiska verktyg. Detta bekräftar även Deissenboeck et al. [16]. De säger att detta borde göras med hjälp av ett eller flera verktyg som kan presentera projektets kvalitet på ett förståeligt sätt. De menar att man med hjälp av kvalitetskontroll kan förhindra att system gradvis förlorar sin kvalitet. De fastställer detta genom att säga:

Obviously, assessing the current state of a system's quality is a key quality control activity.

De säger även att det kan vara svårt att få en klar bild på den nuvarande kvaliteten på ett projekt, men att detta är möjligt genom att man övervakar förändringar över ett tidsintervall. För att lättare kunna identifiera information om trender behövs ett verktyg som bland annat kan lagra och presentera historisk data om projektet och systemet. Ett verktyg som kan användas till detta är Microsoft Team Foundation Server.

2.6 Team Foundation Server 2010

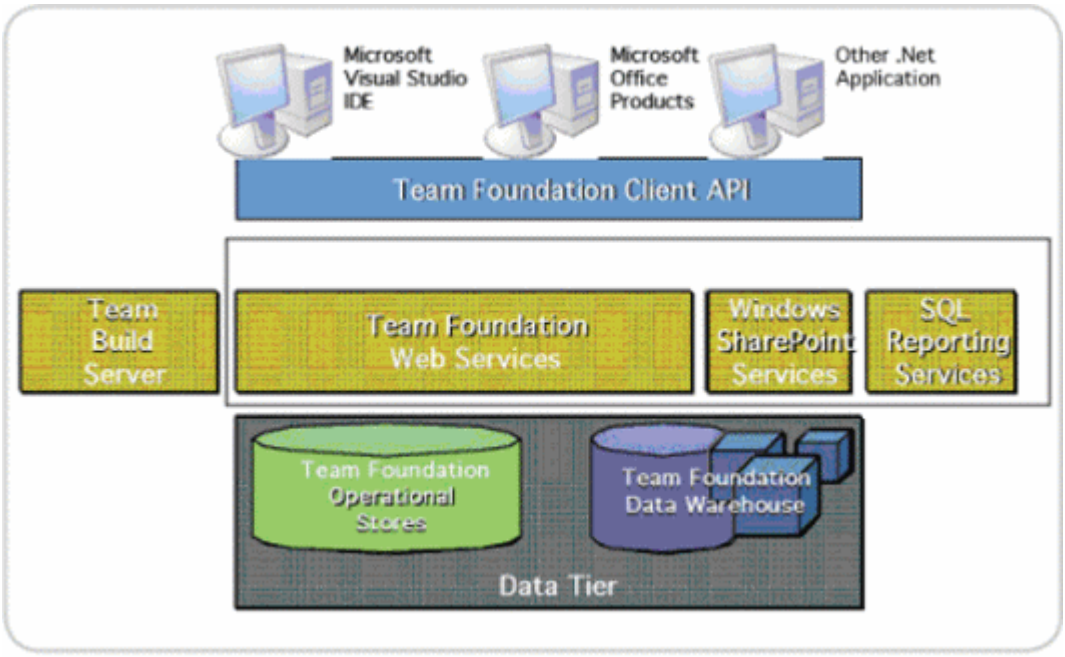
Team Foundations Server 2010 (TFS) är ett system som hjälper systemutvecklingsteam att samarbeta i utvecklingen av IT-system. Det består av ett antal funktioner [1][17] för att stödja denna process:

- **Projekthantering -** Ett projekt kan planeras på en hög nivå i början av projektet genom att man planerar releaser och vilka iterationer som kommer att göras. Man kan även planera ett projekt på en lägre nivå genom att planera vad som ska göras i en iteration. TFS ger också möjligheten att övervaka hur projektet fortlöper.
- **Hantering av Work Items -** Work Items används för att hålla reda på t.ex. buggar, användarkrav, testfall och uppgifter.

- **Versionshantering -** TFS kan användas för att versionshantera bland annat källkod i systemutvecklingsprojekt.
- **Testfallshantering -** Det finns i TFS stöd för att hantera automatiska och manuella tester.
- **Automatisering av byggen -** Det är möjligt att låta TFS automatiskt t.ex. bygga, testa och driftsätta system eller delar av system.
- **Rapportering -** Rapporter kan skapas utifrån data som finns lagrad i TFS. Man kan skapa rapporter för t.ex. buggtrender, iterationsstatus och hur byggen har gått.
- **Hantering av virtuell labbmiljö -** TFS ger möjlighet att skapa och hantera virtuella maskiner. Byggen kan automatiskt driftsättas och testas på dessa. Det är möjligt att spara olika tillstånd på de virtuella maskinerna vilket gör att de senare kan återställas till ett valt tillstånd. Detta underlättar för utvecklarna att återskapa de buggar som de som utför tester hittar.

Team Foundation bygger på en treskiktsarkitektur [18] (se figur 1). Ett klientskikt, ett applikationsskikt och ett dataskikt:

- Team Foundation Server Client API täcker in klientskiktet och interagerar med applikationsskiktet med hjälp av webbtjänster. Det används av Microsoft Visual Studio och kan även användas av andra Windowsapplikationer, som t.ex. Microsoft Excel, för att interagera med applikationsskiktet.
- I applikationsskiktet sker större delen av arbetet som görs i TFS. Det tar bland annat emot anrop från klientskiktet, ger möjlighet till rapportering (SQL Reporting Services) och projektövervakning (Windows Sharepoint Services).
- I dataskiktet sker lagring av information om hela projektet. Detta görs med hjälp av ett antal olika databaser: operativa databaser och ett data warehouse.



Figur 1. En översiktlig bild av arkitekturen i TFS, tagen från MSDN 2011-04-27 [18].

2.6.1 Work Items och processmallar

Data om projekt lagras i TFS i form av Work Items [17]. Vilka typer av Work Items som finns i ett projekt är beroende av vilken processmall som valts. En processmall beskriver för TFS hur ett projekt ska konfigureras och vilka olika komponenter som ska ingå för att stödja en systemutvecklingsprocess. Dessa kan t.ex. vara olika typer av Work Items, dokument och rapporter.

När det på Trafikverkets cfIT skapas nya projekt i TFS 2010 är två mallar vanligast: MSF (Microsoft Solution Framework) for Agile Software Development v5.0 och Microsoft Visual Studio Scrum v1.0. Dessa mallar är väldigt lika varandra. Några av de saker som skiljer dem åt är att Visual Studio Scrum har ett Work Item för Sprints (i princip samma sak som iterationer), att vissa Work Items har olika namn och att attributen i dessa skiljer sig en aning.

De Work Items som finns i dessa mallar (förutom Sprint) har ett antal gemensamma attribut:

- Title - namn på ett Work Item.
- Assigned To - beskriver vem som skapat eller är ansvarig för ett Work Item.
- State - beskriver vad ett Work Item har för status.
- Area - beskriver vilken del i ett system som ett Work Item tillhör.
- Iteration - beskriver vilken iteration ett Work Item tillhör.
- Description - beskriver ett Work Item i text (i Bug finns Steps to Reproduce istället, vilket beskriver hur man ska göra för att reproducera buggen).

2.6.1.1 MSF for Agile Software Development v5.0

MSF for Agile Software Development är en mall som är gjord för att stödja de värderingar som tas upp i det agila manifestet [1]. Alla typer av Work Items kan också länkas till andra Work Items med olika typer av länkar [17].

De Work Items som finns i denna mall är:

- User Story
- Test Case
- Shared Steps
- Task
- Bug
- Issue

User Story används för att beskriva krav från kunden [19]. Den ger en översiktlig bild över en funktion som användare behöver. En User Story ska innehålla tillräckligt med information för att utvecklarna ska kunna uppskatta hur lång tid det kommer att ta för att implementera den. User Story har fyra attribut utöver de gemensamma. Dessa är Reason, Stack Rank, Story Points och Risk.

- Reason används för att beskriva anledningen till att State har det värdet det har.
- Stack Rank beskriver en form av rangordning där utvecklare brukar arbeta på ett Work Item med en låg siffra i detta fält först.
- Story Points beskriver hur mycket arbete som man tror krävs för att implenentera en User Story.
- Risk används för att beskriva vilken riskfaktor en User Story har.
- Acceptance Criteria beskriver vad som ska vara uppfyllt för att en User Story ska anses som klar. Denna beskrivning görs tillsammans med Description.

Test Case används för att definiera antingen ett manuellt eller automatiskt test [20]. Dessa kan sedan hanteras med hjälp av Microsoft Test Manager, där Test Cases kan delas in i Test Suites och testas som grupper. Förutom de gemensamma attributen finns: Priority och Automation Status.

- Priority beskriver vikten av ett Test Case i en skala på 1 till 4, där 1 är den viktigaste.
- Automation Status beskriver om ett Test Case är definierat för ett manuellt eller automatiskt test.

Shared Steps - eftersom många tester använder samma steg i sina utföranden (t.ex. för att logga in på applikationen) kan man med hjälp av Shared Steps definiera en sekvens av steg en gång, och sedan använda det för flera Test Cases [21]**.**

Task används för att beskriva en uppgift som ska utföras [22]. En utvecklare kan till exempel skapa Tasks för att implementera User Stories, och en testare kan skapa Tasks för vissa Test Cases. Förutom de gemensamma attributen finns: Activity, Reason, Stack Rank, Priority, Original Estimate, Remaining och Completed.

- Activity beskriver vilken typ av arbetsuppgift ett Task är skapat för att utföra.
- Reason (se ovan)
- Stack Rank (se ovan)
- Priority beskriver vikten av ett Task i en skala på 1 till 4, där 1 är den viktigaste.
- Original Estimate är de antal timmar man tror ett Task kommer ta att utföra.
- Remaining beskriver den tid som är kvar av den estimerade tiden.
- Completed är antal arbetade timmar som lagts ner på ett Task.

Bug används för att lagra information om de buggar som upptäcks av de som testar systemet [23]. Utvecklarna kan då använda denna information för att fixa buggarna. När en ny bugg registreras sätts dess status till Active. Sedan när buggen har blivit fixad så ändrar man dess status till Resolved. Tester kan sedan utföras för att kontrollera att buggen verkligen är fixad. Om det visar sig att den är fixad sätts dess status till Closed, annars sätts den åter till Active. Förutom de gemensamma attributen innehåller Bug attributen: Reason, Resolved Reason, Stack Rank, Priority och Severity.

- Reason (se ovan)
- Resolved Reason visar vad som är anledningen till att buggen har blivit löst. Dess värde går inte att redigera manuellt utan det uppdateras till det värde som Reason har när State ändras från Active till Resolved.
- Stack Rank (se ovan)
- Priority (se ovan)
- Severity används för att ange hur mycket en bugg påverkar projektet.

Issue beskriver hinder eller saker som måste åtgärdas under nuvarande (eller framtida) iteration [17]. Denna kan resultera i att andra Work Items skapas, t.ex. Bugs eller Tasks. Förutom de gemensamma attributen finns: Activity, Reason, Stack Rank (se ovan för dessa tre) och Due Date (det datum då ett Issue måste vara löst).

2.6.1.2 Visual Studio Scrum v1.0

Visual Studio Scrum [24] är en även det en processmall som är till för att hjälpa till att utveckla system enligt det Agila manifestet. Mallen är väldigt lik MSF for Agile Software Development, men skiljer sig en del när det gäller Work Items och tillgängliga rapporter.

De Work Items som finns i denna mall är:

- Product Backlog Item
- Task
- Test Case
- Shared Steps
- Bug
- Impediment
- Sprint

Product Backlog Item är ungefär samma sak som en User Story, men med vissa skillnader [25]:

- Acceptance Critera anges i ett eget fält i stället för tillsammans med Description.
- Backlog Priority används på ungefär samma sätt som Stack Rank.
- Effort används på samma sätt som Story Points.
- Business Value beskriver vilket värde ett Produkt Backlog Item har för kunden.
- Risk saknas.

Task ser nästan helt likadana ut i de olika mallarna, med vissa olikheter [26]:

- Attributet Block finns med, och beskriver ifall ett Task är blockerat eller inte.
- Backlog Priority finns istället för Priority. Skillnaden mellan dessa är att Priority har ett bestämt värdeintervall medan Backlog Priority kan vara vilken positiv siffra som helst.
- I Visual Studio Scrum finns enbart ett attribut som beskriver en estimering av antal timmar (Remaining Work) arbete som återstår av ett Task. I MSF for Agile Software Development finns möjlighet att estimera total tidsåtgång, antal timmar som återstår och antal arbetade timmar för ett Task.

Test Case skiljer sig inte mellan de olika mallarna [27].

Shared Steps skiljer sig inte mellan de olika mallarna [28].

Bug i Visual Studio Scrum är väldigt lik Bug i MSF for Agile Software Development [29].

- Backlog Priority används i stället för Stack Rank.
- Priority finns inte.
- Effort beskriver hur mycket arbete man tror att det behövs för att fixa en bugg.
- Reason fungerar på samma sätt som i MSF for Agile Software Development men Resolved Reason finns inte.

Impediment är väldigt lik Issue, men de båda skiljer sig en aning från varandra då Due Date och Stack Rank saknas i denna mall [30].

Sprint är ett tillägg som gör att man kan skapa en iteration i form av ett Work Item och på så vis få ut mer information om en sådan [31]. Detta Work Item skiljer sig helt från de andra då dess enda attribut är: Iteration, Start Date, Finnish Date och Sprint Goal.

- Iteration specificerar sökvägen till den iteration som en Sprint tillhör.
- Start Date beskriver när en Sprint börjar.
- Finish Date beskriver när en Sprint slutar.
- Sprint Goal ger en beskrivning av de mål som ska nås i en Sprint.
- Retrospective beskriver resultatet av en sprint t.ex. vad som fungerade bra, vad som fungerade mindre bra och vad som ska göras annorlunda i nästa sprint.

2.6.2 Testning i TFS

Testning är som nämnt en viktig del för att skapa kvalitet på en mjukvara. TFS stöder både manuella och automatiska tester [1]:

- Med manuella tester menas att tester körs på individnivå. Det sker en interaktion mellan en testare och systemet, ofta på samma sätt som slutanvändaren förväntas agera, för att validera funktionalitet och registrera buggar för varje fel som hittas.
- Med automatisk testning menas att man skriver och kör programkod som i sin tur kontroller och granskar mjukvaran som ska testas.

I TFS Test Case Management används Test Plans [32] för att skapa planer för vad som ska testas och hur det ska testas. En Test Plan kan till exempel användas för att planera testningen för en iteration eller en milstolpe i ett projekt. Ett Test Case, som är beskrivningen för vad som ska testas, kan kategoriseras bland andra Test Cases i ett eller flera Test Suites. Vid körning av tester kan man sedan välja att köra specifika Test Cases eller hela Test Suites i olika konfigurationer.

En Test Plan kan innehålla både manuella och automatiska tester. De automatiska tester som stöds är [17]:

- Coded UI test - test som ger stöd för att låta ett användargränssnitt testas för en applikation.
- Unit tests - olika tester som kontrollerar att valda delar av koden för en applikation gör det som de förväntas att de ska göra.
- Database unit test - Unit tests mot databaskod.
- Web performance test - test som kontrollerar funktionalitet eller prestanda för en webbapplikation.
- Load tests - tester som kontrollerar att en applikation uppfyller de krav som ställs på den i form av hastighet genom att utsätta den för olika variationer av belastningar. Dessa tester kan använda webbprestandatester (Web performance tests) eller enhetstester för att stresstesta systemet.
- Generic tests - tester som ger möjlighet att anropa externa testsystem och sedan tolka resultatet från körda tester.
- Ordered tests - tester som innehåller och kör andra tester i en speciell ordning och kan innehålla samma test flera gånger.

2.6.3 Code metrics

Code metrics [33] är ett verktyg som introducerades i Microsoft Visual Studio 2008. Detta verktyg används för att analysera koden och därigenom få information om dess underhållbarhet. När detta verktyg används för att analysera kod beräknas ett antal olika värden:

- **Maintainability Index** - Ett värde mellan 0 och 100 som beskriver hur lätt det är att underhålla koden. Värden mellan 20 och 100 indikerar att koden har god underhållbarhet.
- **Cyclomatic Complexity** - Anger hur strukturellt komplex koden är. Det beräknas utifrån antalet vägar genom koden i programmets flöde.
- **Depth of Inheritance** - Beskriver hur många klassdefinitioner som utökar den klass som ligger högst upp i arvshierarkin.
- **Class Coupling** - Anger hur många andra klasser en klass är beroende av.
- **Lines of Code** - Ungefärligt antal rader kod. Räknar antal rader IL-kod vilket gör att det inte stämmer exakt med antalet rader programkod.

Code Metrics [34] finns numera som kommandoradsverktyg vilket gör det möjligt att köra verktyg utanför Visual Studio. Det i sin tur gör det möjligt att automatiskt köra detta verktyg i samband med byggen och göra så att resultatet sparas i byggloggen. Det är möjligt att konfigurera byggen att misslyckas om resultatet från Code Metrics inte håller sig inom vissa gränser som angivits.

2.6.4 Rapporter

Det finns möjligheter att visa rapporter baserade på de data som finns i TFS [35][36]. Exempel på data kan vara information om Work Items, tester och byggen. Rapporterna kan hämta data från de olika databaserna som finns i TFS. Dessa databaser är olika lämpade för att användas när man skapar rapporter.

De **operativa databaserna** används av de verktyg som finns i TFS och är gjorda för att utföra transaktioner snabbt och ge hög dataintegritet. Tabellerna i de operativa databaserna är normaliserade, vilket gör att man måste utföra många join-operationer för att sammanställa data från dessa. Detta leder till att dessa databaser inte är optimala för att sammanställa data till rapporter eftersom användandet av många join-operationer leder till att exekveringen av frågor tar lång tid och därmed gör att resten av systemet förlorar prestanda.

Det **data warehouse** som finns i TFS är mer lämpat för att sammanställa data till rapporter. Det är en relationsdatabas som är organiserad enligt ett stjärnschema vilket gör det lättare att skriva och exekvera frågor. Driftsättning av TFS data warehouse kan ske på en annan maskin än de operativa databaserna vilket gör att man kan undvika prestandaproblem för de operativa databaserna. Data hämtas från de operativa databaserna med hjälp av adaptrar. Det finns en adapter för varje verktyg i TFS. Data hämtas även periodvis enligt konfigurering.

OLAP-kuben är bra på att analysera historisk data och göra beräkningar. Den hämtar och bearbetar data från det data warehouse som beskrivs ovan. Den kan t.ex. beräkna aggregerade värden i förväg för att det ska gå snabbare att hämta dessa vid senare tillfällen.

Det finns ett antal färdiga rapporter i TFS. Dessa skiljer sig mellan de olika processmallarna. I MSF for Agile Development finns olika rapporter om buggar, byggen, projekthantering och tester [37]. I Visual Studio Scrum finns olika rapporter om byggen, arbete och tester [24].

2.6.4.1 Rapporter i MSF for Agile Software Development

Nedan följer beskrivningar på de rapporter som finns tillgängliga i processmallen MSF for Agile Software Development.

Bug Status Report

Visar en graf över hur antalet buggar, med statusarna Active, Closed och Resolved, har förändrats över tid [38]. Fördelningen mellan olika prioriteter för de buggar som är aktiva visas. För varje medlem i projektgruppen visas hur många aktiva och lösta buggar denne har och dess fördelning mellan olika prioriteter.

Detta krävs för att en Bug Status Report ska vara användbar:

- Buggar måste registreras.
- Status för alla buggar måste anges och uppdateras.
- Prioritet för alla buggar måste anges.
- Den person som ska lösa eller stänga en bugg måste anges.
- Area och Iteration måste anges.

Bug Trends Report

Visar en graf med kurvor för det genomsnittliga antalet buggar som öppnats, lösts och stängts över tid [39]. Värdena för datum i grafen beräknas genom att medelvärdet för de sju föregående dagarna räknas ut.

Detta krävs för att en Bug Trends Report ska vara användbar:

- Buggar måste registreras.
- Status för alla buggar måste anges och uppdateras.
- Area och Iteration måste anges.

Reactivations Report

Visar hur antalet återaktiverade buggar (buggar som blivit satta till active efter att ha varit resolved eller closed) och antalet lösta buggar förändrats över tid i form av en graf [40].

Detta krävs för att en Reactivations Report ska vara användbar:

- Buggar måste registreras.
- Status för alla buggar måste anges och uppdateras.
- Area och Iteration måste anges.

Build Quality Indicators Report

Visar en graf med olika byggen efter x-axeln och flera olika värden på y-axeln [41]. Det finns linjediagram för antal aktiva buggar, hur stor del av koden som täcktes av tester och hur mycket koden ändrats. För varje bygge på x-axeln finns även en stapel som visar hur många tester som lyckats, misslyckats och hur många som varit resultatlösa.

Detta krävs för att en Build Quality Indicators Report ska vara användbar:

- Buggar måste registreras.
- Status för alla buggar måste anges och uppdateras.
- Tester som körs automatiskt vid byggen måste skapas.
- Ett byggsystem måste konfigureras och byggdefinitioner skapas.
- Byggen ska göras regelbundet.

Build Success Over Time Report
Visar en tabell som beskriver en summering av resultat från byggen och tester från ett eller flera projekt över en tidsperiod [42]. För varje kombination av ett bygge, plattform och konfiguration för ett visst datum visas ett resultat. Möjliga resultat är:

- No Build - inget bygge kördes den dagen.
- Build Failed - bygget kördes men misslyckades. Minst ett test misslyckades som inte misslyckats tidigare.
- Build Succeded, No Tests - Bygget lyckades, men inga tester kördes.
- Tests Failed - bygget misslyckades på grund av kompileringsfel eller annat fel.
- Test Passed, Low Coverage - bygget och alla tester lyckades, men mängd kod täckt av testning var låg.
- Passed - bygget och alla tester lyckades och mängden kod täckt av testning var bra.

Detta krävs för att en Build Success Over Time Report ska vara användbar:

- Ett byggsystem måste konfigureras och byggdefinitioner skapas.
- Byggen ska göras regelbundet.

Build Summary Report
Visar en tabell över byggen med kolumner för datum, namn, plattform, konfiguration, hur bygget har gått (Progress), byggets kvalitet (Build Quality), hur stor del av de automatiska testerna som lyckats (% Test Passed), hur stor del av koden som täcks av testerna (Code Coverage) och hur mycket koden har ändrats innan bygget (Code Churn) [43]. Progress kan vara misslyckad, delvis lyckad, lyckad och stoppad. Build Quality är en uppskattning av kvaliteten på ett bygge som gjorts manuellt.

Detta krävs för att en Build Summary Report ska vara användbar:

- Ett byggsystem måste konfigureras och byggdefinitioner skapas.
- Tester som körs automatiskt vid byggen måste skapas.
- Konfigurera tester att sammla data om Code Coverage.
- Byggen ska göras regelbundet.
- Eventuellt utvärdera byggen manuellt.

Burndown and Burn Rate Report
Visar en graf som beskriver antal arbetade timmar (Hours Completed), antal timmar som återstår av den estimerade tiden (Hours Remaining), en linje (Ideal Trend) som beskriver idealisk trend av arbetad tid kontra återstående tid, och en linje (Actual Trend) som visar ett snitt i minskning av tid [44].

Rapporten visar även en graf som beskriver hur många timmar en arbetsgrupp i snitt lägger ner per arbetsdag i en iteration (Actual) och hur många timmar de behöver lägga ner för att hinna klart i tid.

Rapporten använder också en graf som visar hur många timmars arbete projektmedlemmarna har blivit tilldelade, hur många arbetade timmar som gjorts och hur många timmar som finns kvar av de tilldelade timmarna.

Detta krävs för att en Release Burndown ska vara användbar:

- Task, User Stories och Bugs måste definieras.
- Completed och Remaining måste specificeras och uppdateras för varje Task och dess underliggande Tasks.
- Uppdatera State för varje Task, Bug och User Story.

Remaining Work Report
Denna rapport har två vyer, i den ena kan man se hur många timmar arbete som är klara och hur många som återstår [45]. I den andra så kan man se hur många Work Items som är aktiva, klara och stänga. Båda vyerna visar hur dessa värden förändras över tid i form av en graf.

Detta krävs för att en Remaining Work Report ska vara användbar:

- Task, User Stories och Bugs måste definieras.
- Iteration och Area ska anges för dessa Work Items.
- Ange och uppdatera Hours Completed och Hours Remaining för Tasks eller dess underliggande Tasks.
- Uppdatera status på Work Items.

Status on All Iterations Report
Visar en tabell som presenterar det arbete som projektgruppen gjort över flera iterationer [46]. För varje iteration visas:

- Stories Closed - hur många User Storys som blivit avslutade.
- Progress - antal estimerade timmar (Original Estimate), arbetade timmar (Completed) och hur många timmar som återstår av den estimerade tiden för iterationen (Remaining).
- Hur många buggar som är aktiva (Active), löst (Resolved) respektive stängd (Closed).

Detta krävs för att en Status on All Iterations Report ska vara användbar:

- Task, User Stories och Bugs måste definieras och Iteration och Area ska anges för dessa Work Items.
- Ange Original Estimate. Ange och uppdatera Hours Completed och Hours Remaining för Tasks eller dess underliggande Tasks.

Stories Overview Report
Visar en översikt över User Stories i form av en tabell [47]. För varje User Story visas hur stor del av den estimerade tiden som gjorts, hur många timmar som är kvar, test poäng, testresultat och antal buggar som är aktiva och hur många som är lösta. Testpoängen beräknas genom att poängen för de test som testar en User Story summeras. Poängen för ett test är antalet testkonfigurationer som finns för det testet. Testresultat är en visuell och numerisk presentation av hur stor del av testen som vid senaste körningen var lyckade, misslyckade och hur stor del som inte blev körda [48].

Detta krävs för att en Stories Overview Report ska vara användbar:

- Definiera User Stories och Tasks.
- Varje Task ska kopplas till den User Story den tillhör.
- Om det finns sub-Tasks så ska dessa kopplas till sina föräldrar.
- För alla Tasks ska Completed och Remaining fyllas i och uppdateras.
- Buggar ska kopplas antingen till det Test Case som testar buggen med en Tested By länk eller till en User Story med en related To länk.
- När en bugg har blivit fixad ska dess status ändras till Resolved.
- Iteration och Area ska anges för Task, Test Case och Bug.

Stories Progress Report
Visar en tabell som ger översikt på hur mycket arbete som skett för ett antal User Stories och dess underliggande Tasks [49]. Den visar titeln för en User Story, följt av ett procentuellt värde av hur mycket arbete som lagts ner (tillsammans med en mätare som visar Completed, Recently Completed och Hours remaining), och sist antal timmar fortfarande tillgängliga för denna.

Detta krävs för att en Stories Progress Report ska vara användbar:

- User Stories och Tasks måste definieras.
- En Child-länk måste skapas från varje Task till den User Story som den implementerar. En Child-länk måste även skapas från underliggande Task till det som implementeras.
- Ange och updatera Hours Completed och Hours Remaining för Tasks eller dess underliggande Tasks.
- Iteration och Area ska anges för varje Work Item.

Unplanned Work
Visar en graf som presenterar antal Work Items som skapats före en iterations startdatum (Planned) och de antal Work Items som klass som "oplanerade" och som skapats efter detta datum [50].

Detta krävs för att en Unplanned Work Report ska vara användbar:

- När en User Story, Task, Test Case eller Bug skapas måste Iteration anges.

Test Case Readiness Report
Visar en graf som beskriver hur många Test Cases som har status satt till Design eller Ready [51].

Detta krävs för att en Test Case Readiness Report ska vara användbar:

- Test Cases måste definieras.
- Iteration och Area måste anges för dessa.
- Status för ett Test Case ska uppdateras.

Test Plan Progress Report
Visar hur många testpoäng som hör till kategorierna Passed, Failed, Blocked, Never Run och Other [52]. testpoängen för en kategori räknas ut genom att antalet testkonfigurationer som hör till denna kategori för varje testfall summeras [48].

Detta krävs för att en Test Plan Progress Report ska vara användbar:

- Test Cases måste definieras.
- Iteration och Area måste anges för dessa.
- Test Plans ska definieras och status för dessa anges.
- För varje manuellt test ska alla teststeg markeras som Passed eller Failed. För automatiska tester sker detta automatiskt.
- Area och Iteration ska anges för varje Test Case.

2.6.4.2 Rapporter i Visual Studio Scrum

Nedan följer beskrivningar på de rapporter som finns tillgängliga i processmallen Visual Studio Scrum.

Release Burndown
Visar en graf som beskriver hur mycket tid som återstod av den totala estimerade tiden för en Release i början av en Sprint [53]. Rapporten går att filtrera på Area eller Sprint.

Detta krävs för att en Release Burndown ska vara användbar:

- Product Backlog Items och Bugs måste vara definierade, Iteration och Area specificerade.
- Effort måste specificeras och uppdateras för varje Product Backlog Item och Bug som arbetas på.
- Uppdatera State för varje Product Backlog Item och Bug från New till Done när de är färdiga.

Sprint Burndown
Visar en graf över ett specificerat tidsintervall i en Sprint och beskriver hur mycket arbete (i timmar) som finns kvar för en Sprint (To Do), hur stor del som arbetas på just nu (In progress) och en idealisk trend som visar en konstant minskning i antal estimerade arbetstimmar som finns kvar [54]. Rapporten går att filtrera på Area eller Sprint.

Detta krävs för att en Sprint Burndown ska vara användbar:

- Task måste vara definierade, och Iteration och Area specificerade för detta.
- Remaining Work måste specificeras och uppdateras.
- State måste uppdateras för varje Task.

Velocity
Visar en graf över flera Sprints och den arbetskraft (Effort) som använts för var och en av dessa.

Detta krävs för att en Velocity Report ska vara användbar:

- Product Backlog Items och Bugs måste vara definierade, Iteration och Area specificerade.
- Effort måste specificeras och uppdateras för varje Product Backlog Item och Bug med State satt till Active.
- State måste uppdateras för varje Product Backlog Item och Bug [55].

Visual Studio Scrum har 4 rapporter gemensamma med MSF for Agile Software Development:

- Build Success Over Time Report
- Build Summary Report
- Test Case Readiness Report
- Test Plan Progress Report

2.6.4.3 Egna rapporter

Man kan även skapa egna rapporter och modifiera de existerande [17]. Rapporter kan skapas med vilket verktyg som helst som kan ansluta till ett data warehouse eller en analysdatabas. Två typer av rapporter som man kan skapa från TFS är Excel-rapporter och RDL-rapporter (Report Definition Language). För att skapa Excel-rapporter kan man antingen använda Work Item queries eller Microsoft Excel. För att skapa RDL-rapporter så kan man använda Report Designer och Report Builder. Om man använder Work Item queries jobbar man mot TFS operativa databaser. De andra alternativen jobbar mot antingen TFS data warehouse eller OLAP-kuben.

2.6.4.4 Project Portals och Dashboards

En Project Portal är en webbsida som skapas på en Sharepoint-server för varje projekt som skapas och fungerar som en informationsportal [17]. På denna visas bl.a. dokumentation om projektet, rapporter och det går att hantera så kallade Dashboards.

En Dashboard är kort sagt en samling av rapporter. Projektmedlemmarna kan välja en av flera Dashboards, och utifrån en sådan välja ett antal rapporter för att få information om t.ex. projektets kvalitet eller hur testningen går. För att kunna visa detta krävs läsrättigheter till projektet, och för att kunna göra ändringar måste man ha rättigheter att redigera.

TFS har sex stycken inbyggda Dashboards:

- Progress (Burndown)
- Quality
- Bugs
- Test
- Build
- Project
- My Dashboard

Antalet Dashboards som är möjliga att använda är beroende av vilken Sharepoint-server man använder [56]. För att kunna använda alla måste man Microsoft Office Sharpoint Server 2007 Enterprise Edition. I andra fall är enbart My Dashboard och Project Dashboard tillgängliga.

Var och en utav de ovan nämnda Dashboards går att ändra på, och det går även att lägga till nya Dashboards för att kunna övervaka eller följa upp projektet med annan information som grund [17].

2.7 Sammanfattning

En triangel av produktkvalitet, tid och resurser kan användas för att beskriva kvaliteten på ett utvecklingsprojekt. Tanken med triangeln är att förändring av en av aspekterna påverkar de andra två, och att de därför måste balanseras. Flera författare argumenterar dock för att projekt ska anpassas efter kunden och användarnas krav, som t.ex. att produkten har god användbarhet och att den blir klar i tid. Produktkvalitet kan beskrivas med kvalitetsattributen: funktionalitet, pålitlighet, användbarhet, effektivitet, underhållbarhet, portabilitet och återanvändbarhet.

För att uppnå kvalitet i de tre aspekterna kan man använda sig av agila utvecklingsmetoder där fokus ligger på bland annat kundsamarbete och anpassning till förändringar. I dessa metoder är testning viktigt.

Det är viktigt att mäta framsteg för att projektet ska hålla sig inom tidsramen och för att produkten ska hålla hög kvalitet. Detta kan göras genom att övervaka förändringar över olika tidsintervall. Det behövs då verktyg som kan spara och presentera historisk data om ett projekt. Ett verktyg som stödjer detta är Team Foundation Server 2010 (TFS).

TFS är ett system som är till för att ge stöd för systemutvecklingsprojekt. Projekt skapas i TFS utifrån en vald processmall. Dessa processmallar definierar ett antal Work Items som beskriver olika typer av arbetsuppgifter som kan uppstå i ett projekt. Det är möjligt att, i ett projekt, skapa nya och redigera existerande Work Items för att anpassa TFS-projektet efter speciella behov. Utifrån dessa kan man även, tillsammans med resultat från byggen och tester, skapa rapporter. Vilka rapporter som finns beror på vilken processmall som valts, men det är även möjligt att skapa nya rapporter och redigera befintliga.

3 Metod

Det finns mycket skrivet om hur man kan utvärdera kvalitet på systemutvecklingsprojekt, men inte om hur man kan göra detta med hjälp av mätetal från TFS. Det finns redan färdigställda rapporter i TFS för att visa ett antal av de tillgängliga mätetalen. Men är dessa tillräckliga för att ge en bra överblick av kvaliteten på ett systemutvecklingsprojekt och vilka av dessa mätetal i TFS kan användas för detta?

3.1 Strategi och datainsamling

3.1.1 Kvalitativa och kvantitativa metoder

I kvantitativa metoder samlas mätbar data utifrån en eller flera företeelser. Dessa data kan sedan analyseras genom att olika variabler jämförs för att hitta samband mellan dessa. Detta kan göras med hjälp av t.ex. statistiska beräkningsmodeller. Med kvantitativa metoder kan det alltså hittas samband mellan olika företeelser, men inte visas varför dessa existerar. I kvalitativa metoder däremot, studeras företeelser i detalj. Fokus ligger i att ta reda på varför samband mellan företeelser existerar [57].

3.1.2 Val av metod

För att kunna hitta sambandet mellan mätetal från TFS och kvaliteten på systemutvecklingsprojekt med hjälp av kvantitativa metoder krävs det att man kan mäta dessa. Ett sätt kan vara att man mäter kvaliteten på olika systemutvecklingsprojekt med hjälp av t.ex. en enkät för att sedan jämföra resultatet med mätetalen från TFS för dessa projekt, och på så sätt hitta ett samband. Detta skulle dock vara svårt och omständigt, eftersom det går att kombinera mätetal från TFS på många olika sätt utöver de som används i rapporter och även skapa nya. Alternativet är att använda en kvalitativ metod för att hitta sambandet. Detta kan göras genom att man undersöker hur kvaliteten kan härledas från mätetal med hjälp av att personer som är insatta i området intervjuas. Detta är den metod som kommer att användas för detta examensarbete.

Denna fallstudie kommer att göras för Trafikverkets centrala funktion IT. På avdelningen där examensarbetet utförs finns ett antal projektledare och systemutvecklare. Ca 10st av dessa personer kommer att väljas ut genom att Trafikverkets handledare för detta examensarbete ger förslag på personer som är insatta i Team Foundation Server. Dessa kommer sedan att få svara på ett antal frågor i en intervju som spelas in (de intervjuade måste godkänna inspelning och kommer att vara anonyma). Målet är att hälften av personerna ska vara utvecklare och hälften projektledare. Detta för att få synpunkter från både ledning och utveckling.

För att de som ska intervjuas ska vara förberedda och lättare kunna svara på frågor om mätetal, kommer en lista över mätetal att skickas ut i förväg. Mätetalen i denna lista är härledda från de rapporter som ingår i de processmallar som beskrivs i avsnitt 2.5.1, och kommer att vara indelade efter Work Items, iterationer/sprints och byggen för att göra listan mer lättöverskådlig.

De frågor intervjuerna baseras på kommer att kretsa kring triangeln av produkt-kvalitet, tid och resurser. För varje del ställs frågor om vilka data som krävs för att utvärdera den, vilka mätetal från TFS som kan användas för detta, hur de kan användas och om de är tillräckliga. Frågorna som kommer att användas vid intervjuerna finns i bilaga 1. Eventuella följdfrågor kan tillkomma för att få ytterligare klargörande om de intervjuades åsikter i de fördefinierade frågorna. När frågor som behandlar produktkvalitet ställs kommer en lista över kvalitetsattributen från ISO/IEC 9126, utökat med återanvändbarhet, att visas. Detta för att den intervjuade ska kunna reflektera kring hur viktiga dessa är och om det är möjligt att uppskatta dem utifrån TFS.

3.2 Analys av resultat

Sammanställningen av intervjuerna kommer att ske genom att inspelningarna lyssnas igenom och de intervjuades åsikter kring de olika frågorna noteras. Åsikter kommer att grupperas efter vilken fråga de berör och efter vilken typ av åsikt det är (t.ex. positiv eller negativ till att använda triangeln för att beskriva projektkvalitet). De kommer att färgkodas för att markera vilken intervju de kommer ifrån. Sedan kan varje fråga studeras för att identifiera vilka åsikter som är vanligast. Även avvikande åsikter kommer att presenteras för att ge en komplett bild av de åsikter som förekommer.

3.3 Begränsningar och potentiella problem

Eftersom de olika mätetalen i TFS kan kombineras på väldigt många olika sätt är det svårt att kartlägga alla dessa inom tidsramen för detta examensarbete. Därför har endast de kombinationer som finns i de rapporter som nämns i tidigare avsnitt kartlagts. Eventuella andra kombinationer av dessa mätetal kan komma upp i samband med intervjuerna.

Resultatet av frågorna kan komma att variera med tanke på det faktum att personerna i intervjuerna kan ha olika mycket erfarenhet av hur TFS bör eller kan användas. Det kan även vara beroende av intervjuarnas erfarenhet inom kvalitativ datainsamling.

4 Resultat

Det här kapitlet presenterar resultatet av den fallstudie som beskrivs i avsnitt 3 – Metod. Av de personer som intervjuades var sex systemutvecklare och tre projektledare. Kunskaper kring TFS skiljde sig mellan dessa vilket gjorde att vissa intervjuer gav mer konkret information kring mätetal än andra. Resultatet presenteras indelat efter de fem huvudaspekterna som behandlades under intervjuerna (se bilaga 1 för intervjufrågor).

Något som upptäcktes allt eftersom intervjuerna utfördes var hur TFS används i olika projekt på Trafikverket. I vissa projekt använder man alla Work Items, men är osäker på hur dessa ska användas. I andra projekt används inte Work Items över huvud taget. Detta bör tas i åtanke vid bedömandet av resultatet och i vidare diskussion kring ämnet.

4.1 Triangeln

Alla intervjuer inleddes med att ifrågasätta ifall triangeln är tillräcklig att använda för att ge en beskrivning av projektkvalitet, där samtliga intervjuade ansåg att triangeln stämmer överens med hur man ser på projektkvalitet. Flera påpekade dock att prioritering av delarna i triangeln skiljer sig mellan olika projekt.

4.2 Sammanvägning

De flesta av de intervjuade trodde att det kan vara möjligt, men svårt, att väga samman mätetal kring tid, resurser och produktkvalitet till ett värde som beskriver hälsan på ett projekt. Projektledarna var överlag positiva till möjligheten av en sammanvägning.
Systemutvecklarna nämnde problem som t.ex. att ändrade förutsättningar kan göra att viktningen kan ändras under projektets gång och att olika projekt kräver olika viktning.

4.3 Produktkvalitet

Det rådde lite delade meningar kring om mätetal från TFS är tillräckligt för att ge en bra bild av produktkvaliteten. De som var mindre positiva till denna möjlighet trodde dock att de kanske kan ge en avspegling av produktkvaliteten.

En syn på produktkvalitet som återkom i många intervjuer var att den är tätt kopplad till hur slutanvändarna upplever produkten. Förslag på hur man kan utvärdera detta var användartester kombinerat med enkäter eller intervjuer.

Majoriteten ansåg att man kan utvärdera en produkts pålitlighet med hjälp av TFS. En idé som flera delade var att man kan utgå ifrån mätetal kring buggar och hur stor del av koden som täcks av tester (enhetstester nämndes i flera intervjuer). De menade att om det finns få buggar och stor del av koden täcks av tester har produkten hög pålitlighet, men om det däremot finns mycket buggar eller om en liten del av koden täcks av tester är pålitligheten lägre. En av de intervjuade ansåg att information om buggar är mindre användbart och förespråkade i stället att man använder information om testfall.

Mätetal kring byggen, som testresultat och hur mycket koden har förändrats sedan föregående bygge (Code churn), togs upp av flera av de intervjuade. De var överens om att hög andel lyckade tester tyder på hög kvalitet. De som talade om Code churn ansåg att den ska vara låg, alltså att koden inte ändras mycket, eftersom förändringar kan medföra nya buggar vilket leder till sämre kvalitet. En annan åsikt om Code churn var att den ska vara ganska hög i början av ett utvecklingsprojekt eftersom det då ska skapas mycket ny kod.

Många ansåg att det är viktigt att mäta underhållbarheten på koden, och att kod-komplexitet är en egenskap som påverkar denna. Flera tyckte att man kan använda Code metrics för att ta reda på kodkomplexitet. Andra sätt för att utvärdera underhållbarhet som nämndes var att undersöka hur lång tid det tar att rätta buggar av olika svårighetsgrad (Severity) eller hur lång tid det tar att få ändringar driftsatta.

En av de intervjuade funderade kring effektivitet och undrade om det kanske är möjligt att göra prestandatester i samband med byggen. Denne nämnde att man skulle kunna mäta hastigheten på databasfrågor i SQL Server.

Portabilitet var ett kvalitetsattribut som det talades väldigt lite om under intervjuerna. Det var dock en som påpekade att det kan vara intressant i vissa projekt. Han påpekade även att detta inte är något som man kan få ut information om från TFS.

4.4 Tid

En gemensam åsikt hos många var att det utifrån TFS är lättare att mäta tidsaspekten än produktkvalitet och resurser. Enligt vissa är tidsplanering och tidsuppföljning den viktigaste delen för projektkvalitet, men de flesta ansåg att det varierar beroende på hur man prioriterar i ett projekt.

Majoriteten av de intervjuade tog upp att Task och iterationer/Sprints borde användas för visa hur man tidsmässigt ligger till i ett projekt. Detta genom att jämföra den mängd tid som lagts ner för Tasks med den estimerade tiden för dessa, och även jämföra dessa med milstolpar för en Sprint/iteration. Det kan i sin tur visas med hjälp av en Burndown-rapport, som av flera ansågs vara en av de bästa rapporterna som är Tillgänglig i TFS.

Flera påpekar att man borde använda en rapport i stil med Unplanned work för att se hur mycket oplanerat arbete som tillkommit under en iteration/Sprint. Detta för att det i agila utvecklingsprojekt ofta tillkommer arbete efter planeringsfasen för en iteration/Sprint. Det blir då viktigt att jämföra milstolpar och tidsplan med arbetsuppgifter som tillförts i planeringsfasen i jämförelse med arbete som tillkommit efter iterationens startdatum. Ett antal personer tycker även att det kan vara intressant om man skulle kunna få någon form av information om arbete från hela projektet istället för enbart iterationer/Sprints.

4.5 Resurser

Vissa menade att man kan ta reda på om arbetet är bra fördelat över projektmedlemmarna genom att undersöka hur t.ex. buggar är fördelade med hänsyn till deras svårighetsgrad (Severity). Några menar dock att Work Items inte ger en korrekt bild av arbetsfördelningen t.ex. eftersom flera kan samarbeta för att lösa buggar och göra Tasks. Flera påpekade att man i TFS inte kan se hur mycket olika projektmedlemmar jobbar i ett projekt vilket också det gör det svårt att utvärdera fördelningen av arbete. En av de intervjuade ansåg att man inte delar ut en uppgift till en person i ett projekt som inte har tid för det och att det därför är mindre intressant att utvärdera fördelningen av arbetet.

Några av de intervjuade konstaterade att resurser är starkt kopplat till tidsaspekten. En menade att man borde se till resurserna på team-nivå i stället för individnivå. Några påpekade att antalet projektmedlemmar oftast är konstant och att tiden eller produktkvaliteten/omfattningen anpassas. Ett antal av de intervjuade tyckte att Burndown-rapporter kan användas för att avgöra om man har tillräckligt med resurser. Detta eftersom tid och resurser är nära kopplade till varandra, och att man genom att se till hur man ligger till i mängden arbete utfört i jämförelse med mängden arbete kvar kan se om man har tillräckligt med resurser eller inte.

4.6 Sammanfattning

Alla de intervjuade personerna var överens om att triangeln kan användas för att beskriva projektkvalitet, men däremot rådde det skilda meningar om det går att väga samman produktkvalitet, tid och resurser för att ge ett värde på kvaliteten för ett projekt.

Produktkvalitet anses av de flesta vara starkt kopplad till hur användaren upplever produkten och att det därför bör utvärderas genom kontakt med slutanvändaren. Vissa delar av produktkvaliteten ansåg dock de flesta att man kan utvärdera med hjälp av mätetal från TFS. De som förespråkades mest var mätetal kring buggar i kombination med testfall och information om byggen, t.ex. antal rader kod som ändrats, testresultat och Code metrics.

Flera ansåg att resurser och tid har en stark koppling och att det på ett bra sätt går att få ut bra mätetal om tid, men att det är mer tveksamt hur man ska använda TFS för att beskriva användning av resurser i ett projekt. Estimerad tid, använd tid och tid kvar utifrån Task är viktiga mätetal för tidaspekten, och även mängd arbete som tillkommit. Ett förslag som många menar är det bästa alternativet för att utvärdera både tid och resurser är att man jämför mängd arbete som lagts ner i jämförelse med arbete som måste göras tillsammans med milstolpar (Burndown-rapporter).

5 Diskussion

5.1 Triangeln

Eftersom alla de intervjuade samt författarna av den litteratur som studerats är överens om att triangeln är lämplig för att beskriva kvalitet på utvecklingsprojekt kan man se det som en allmän åsikt. Det borde dock tas i åtanke att denna triangel är en gammal modell, och att den av vissa inte anses helt fulländad. För detta examensarbete krävdes dock en utgångspunkt och en mindre avancerad modell för att kunna lägga fokus på hur TFS kan användas för att ge en enkel men bra bild av projektkvalitet.

5.2 Produktkvalitet

För att man ska kunna diskutera kring produktkvalitet är det viktigt att man får en komplett bild av vad det är. Man kan tycka att om man använder en standard, som t.ex. ISO/IEC 9126, för att definiera produktkvalitet borde det ge en komplett bild över detta. Det kan dock diskuteras om denna ska användas för att ge en helhetsbild av kvaliteten på ett system. Det kan finnas nya standarder som tillkommit sedan denna skapades som bättre beskriver detta. Dock är kvalitetsattributen i ISO/IEC 9126 allmänt accepterade och kan argumenteras ge en tillräckligt bra beskrivning av produktkvalitet för att den ska kunna användas för att ge en översiktlig bild av detta.

5.2.1 Återanvändbarhet eller inte?

Som framgick i den teoretiska bakgrunden anser Dromey att ISO/IEC 9126 bör utökas med kvalitetsattributet återanvändbarhet. Det kan diskuteras om detta verkligen är nödvändigt, då återanvändbarhet till stor del påverkar underhållbarhet och därmed kanske inte borde vara ett eget kvalitetsattribut. Men det kan också vara som Dromey påstår, att inte alla delar i detta attribut påverkar underhållbarheten. Vi anser att eftersom de flesta delarna i återanvändbarhet påverkar underhållbarhet borde man därför kunna bortse från Dromeys tillägg, i alla fall när man endast vill ge en översiktlig bild av produktkvalitet.

5.2.2 Användbarhet och funktionalitet

Dromey anser också att man kan använda sig av kodanalys för att avgöra de olika kvalitetsattributen för ett system. Detta kan vara en aning motsägande vad gäller användbarhet och funktionalitet. Om man ser till de intervjuades åsikter och det agila synsättet, som säger att det är kundens eller användarens åsikt som avgör dessa attribut, borde man istället använda sig av sätt som tar hänsyn till användarens krav. Detta skulle kunna göras med hjälp av användartester i kombination med antingen intervjuer eller enkäter. En variant kan också vara att man sparar data från sådana tester i TFS och använder dessa som mätetal. Det kan även diskuteras ifall kodade gränssnittstester (Coded UI tests) kan användas för att få ett svar på hur bra användbarhet systemet har. Dessa ger dock bara svar på om systemet gör det utvecklarna förväntar sig att det ska göra, och tar inte hänsyn till användarens krav.

Underlag för utvärdering av funktionalitet skulle kunna skapas automatiskt i TFS om man skapar kodade UI-tester. Detta kan dock komma att kräva merarbete eftersom kraven i agila systemutvecklingsprojekt ska kunna förändras under projektets gång. Det kan tänkas att enhetstester kan användas för att mäta om systemet fungerar som det ska. Dessa kan dock förmodligen endast bevisa att ett system inte fungerar som det ska eftersom de endast testar små delar av koden och inte t.ex. att användargränssnittet fungerar. Man skulle även kunna använda manuella tester för att testa funktionaliteten. Dessa kräver dock mer arbete vid utförandet av testerna relativt kodade UI-tester men däremot mindre för att skapa och underhålla dem. Det kan vara svårt att avgöra

generellt om man ska använda kodade UI-tester eller manuella tester eftersom vilket alternativ som är effektivast kan bero på hur man arbetar.

5.2.3 Effektivitet

I en intervju nämns det att det skulle vara bra att mäta prestanda, och att det skulle kunna göras med hjälp av att mäta hastigheten på databasfrågor i SQL Server. Då det bara är en av de intervjuade som nämner prestanda som en viktig del i produktkvalitet kan även vikten av denna diskuteras. Man skulle kunna anta att de andra inte tycker att detta är en viktig del i produktkvalitet. Det mer troliga alternativet är dock att de inte är medvetna om det är möjligt att få stöd för att få information om effektivitet ifrån TFS och därför valde att inte ta upp detta kvalitetsattribut. Det går dock att anse att detta attribut fortfarande är en viktig del i utvärderingen av en produkts kvalitet.

Ett alternativ för att utvärdera effektiviteten av ett system skulle kunna göras med hjälp av stresstester (Load tests). Det krävs dock att man har skapat enhetstester eller webbprestandatester för att dessa ska kunna skapas. Troligtvis har man redan skapat enhetstester och därmed krävs inte allt för mycket arbete för att skapa denna typ av test.

5.2.4 Portabilitet

En av de intervjuade nämnde att det kan vara intressant att utvärdera portabiliteten i vissa projekt, men att det inte går att få ut information om detta från TFS. Ingen av de andra nämnde dock detta kvalitetsattribut. Det kan därför diskuteras om detta är viktigt, eller om man kan bortse från det när man vill ge en översiktlig bild av produktkvalitet. Eftersom det troligen är intressant att utvärdera i vissa projekt och att TFS inte ger stöd för detta kan det argumenteras för att vinsten inte väger över det arbete som krävs för att kunna utvärdera detta.

5.2.5 Pålitlighet

De flesta av de mätetal som de intervjuade talade om när de fick frågor om produktkvalitet kan anses vara sådana som påverkar pålitligheten. Detta skulle kunna bero på att det är det kvalitetsattribut som har flest uppenbara mätetal i TFS. Alternativt kan det vara så att det är sådana mätetal som personer inom mjukvaruutveckling relaterar till när de tänker på produktkvalitet. Kanske hade resultatet sett annorlunda ut om de intervjuade inte visste att det handlade om mätetal från TFS.

Antal aktiva buggar i kombination med en siffra på hur stor del av koden som är täckt av tester var en kombination av mätetal som många pratade om. Tanken är att man ska ha få buggar och att stor del av koden ska vara täckt av tester för att ha hög pålitlighet. Ensamma kan dessa enbart visa att systemet har dålig pålitlighet, t.ex. om det visar sig att man har många buggar eller att en liten del av koden täcks av tester. Om man har väldigt få buggar kan det antingen bero på att det är ett buggfritt system eller att man inte har testat tillräcklig mycket. Om man istället jämför de båda kan man både få svar på om man testat tillräckligt och om testerna påvisar ett buggfritt system.

I intervjuerna togs det upp att mätetal kring byggen kan vara viktiga. Ett av dessa var andel tester som lyckats. För att detta ska vara ett intressant mätetal behöver man jämföra detta med hur stor del av koden som täcks av tester. Detta kan motiveras med ett fall där man har 100 % lyckade tester men endast har ett test som täcker 1 % av koden. Ser man bara till andel lyckade tester kan man lätt tro att det är hög kvalitet på systemet, men tas det hänsyn till hur stor del av koden som täcks av tester inser man snabbt att detta troligtvis inte är fallet.

Ett annat mätetal kring byggen som föreslogs under intervjuerna var antal rader kod som förändrats innan ett bygge (Code churn). Det kan diskuteras hur detta påverkar ett system. Om många rader kod har förändrats kan det påverka kvaliteten på koden negativt, då det kan tillkomma nya buggar, men det kan även påverka kvaliteten

på systemet positivt, då koden kan ha förbättras i olika avseenden. Man kan då fundera kring hur mycket detta egentligen säger om kvaliteten på koden. Det kan ju vara så att koden som ändrats täcks av tester som körts i samband med bygget. Om dessa tester har lyckats kan det betyda det att koden inte har blivit mindre pålitlig på grund av förändringarna. Detta argumenterar för att man bör ta detta mätetal i jämförelse med hur stor del av koden som täcks av tester och hur stor del av dessa tester som lyckats. Man kan då diskutera om detta mätetal tillför något i jämförelse med om man endast använder andel lyckade tester i jämförelse med hur stor del av koden som är täckt av tester. Förändring av kod kan ses som en orsak till att kvaliteten på systemet förändrats och skulle därför kunna användas för att ge en förklaring till varför kvaliteten förändrats.

Vilka mätetal ska man då använda för att visa pålitligheten för ett system? Två olika kombinationer av mätetal har visat sig vara användbara för att visa detta: andel kod som är täckt av tester i kombination med antingen antal aktiva buggar eller andel tester som lyckats. Frågan är då om man bara ska använda en av kombinationerna eller båda. Testresultat och buggar är ganska nära kopplade till varandra eftersom misslyckade tester kan leda till att en eller flera buggar hittas. Det är dock även möjligt att en bugg får flera tester att misslyckas. Man kan säga att testresultat visar hur stor del av systemet som inte fungerar och att buggar visar hur många fel det finns i systemet. Vilken av dessa kombinationer som är mest intressant är därför beroende på om man är intresserad av att veta hur mycket som behöver rättas till för att få systemet att fungera eller om man är intresserad av hur stor del av systemet som fungerar. Många påpekade att användaren och kundens syn på produktkvaliteten som är viktigast och därför kan det kanske skilja sig mellan olika projekt vilket av dessa alternativ som är mest intressant. Det är troligt att man vill använda en kombination av dessa alternativ och även se till hur mycket kod som ändrats inför det senaste bygget för att få en bra bild av pålitligheten på systemet och få insikt i varför den är hög eller låg.

5.2.6 Underhållbarhet

I några av intervjuerna tas det upp att Code Metrics, mätning av hur lång tid det tar att rätta buggar av olika svårighetsgrad (Severity) och hur lång tid det tar att få ändringar driftsatta, skulle kunna användas för att visa ett systems underhållbarhet. Men hur kan man på bästa sätt använda dessa?

Code Metrics ger svar på hur komplicerad koden är, vilket gör detta till ett bra alternativ eftersom kodkomplexitet i hög grad påverkar hur underhållbar koden är. Om koden t.ex. har låg komplexitet kan det vara enklare att göra ändringar eftersom det är lättare att förstå koden. Detta gör att man snabbare kan komma fram till vad som behöver ändras och risken för oväntade resultat minskar.

Hur lång tid det tar att rätta buggar av olika svårighetsgrad eller tiden det tar att få ändringar driftsatta är två alternativ. Detta alternativ kräver förmodligen att man jämför med andra projekt eftersom tiden det tar troligtvis inte säger något om man inte har något att jämföra med. När man ska utvärdera underhållbarheten för ett system är det inte säkert att man har möjlighet att hämta data från andra projekt. Det kan vara nödvändigt att man analyserar flera projekt och tar fram en skala som kan användas för att utvärdera dessa mätetal. Det är dock troligt att denna metod är för osäker i förhållandet till det arbete som krävs för att implementera den. Det kan argumenteras för att Code Metrics är ett bättre alternativ då det är ett färdigt verktyg som kan köras automatiskt i samband med byggen.

5.3 Tid och resurser

Informationen man kan få ut om resurser från TFS är väldigt begränsad. Det alternativ som diskuterats under intervjuerna är att mäta fördelningen av arbete genom att undersöka hur t.ex. buggar och tasks är fördelade på olika personer. Problem som har nämnts är att Work Items endast tilldelas en person men att flera kan jobba med dem och att man inte genom TFS kan få reda på hur mycket olika personer jobbar i ett projekt. Detta gör att bilden av den arbetskraftsfördelningen inte stämmer överens med verkligeheten. Det kan därför diskuteras om dessa mätetal borde användas.

Eftersom det i intervjuerna framgick att resurser ofta är relativt konstanta i projekt, och att det visar sig att tid och resurser anses ha en stark koppling, skulle en lösning till problemet kring resurser därför kunna vara att man mäter tidsaspekten för att på så vis se om det finns tillräckligt med resurser tillsatta i ett projekt.

I intervjuerna var ett alternativ för att utvärdera tid det som ansågs bättre än alla andra. Det var att, för alla Tasks i en Sprint/iteration, studera den estimerade, återstående och den arbetade tiden för att ta reda på om man hinner klart med alla Tasks som ska göras innan Sprinten/iterationen är slut. Detta visas i TFS med Burndown-rapporter. En annan rapport som de intervjuade anser borde användas är Unplanned Work. Denna rapport visar hur mycket arbete som tillkommit under en Sprint/iteration. När det gäller frågan om man hinner klart med arbetet i tid kan man argumentera för att denna rapport är onödig eftersom resultatet av tillagt arbete även syns i Burndown-rapporten. Det kan dock vara intressant att ur ett utvärderings-perspektiv kunna se hur mycket arbete som tillkommit under en Sprint/iteration och då kan denna rapport vara användbar. Den skulle även kunna användas för att övervaka att man inte tar åt sig allt för mycket ändringar i slutet av en Sprint/iteration.

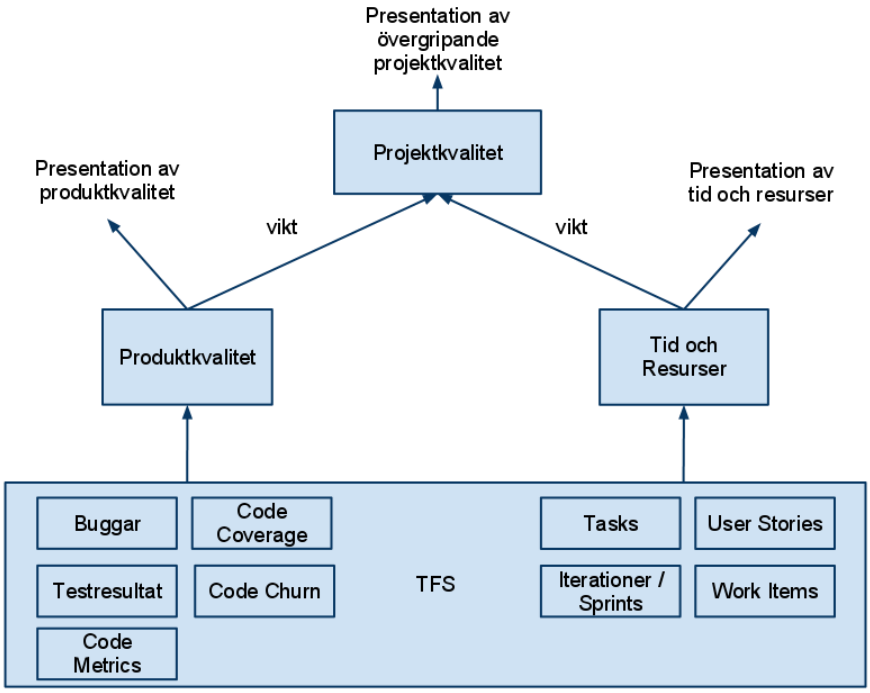
Som ett antal personer påpekade under intervjuerna kan det vara intressant att utvärdera tidsaspekten ur ett projektperspektiv och inte bara se till den pågående iterationen/Sprinten. Eftersom man inte från början planerar allt arbete i detalj, kan det dock vara svårt att utifrån Task utvärdera om man ligger bra till i tidsplanen. Kanske är det mer lämpligt att använda User Stories/Product Backlog Items för detta. Det är dock inte säkert att man har alla sådana uppskattade, och dessa kan ändras under projektets gång. Man skulle kunna se till tidigare slutförda Sprints/iterationer för att få någon typ av bild över hur projektet ligger till i tidsplanen. Detta ger dock en begränsad bild, eftersom den inte tar hänsyn till hur mycket arbete som återstår i projektet. Det kan t.ex. vara så att man kör fem lyckade sprintar men att det visar sig att man har mycket arbete kvar ändå nära projektets deadline.

5.4 Sammanvägning

Som framgick i avsnitt 4 (Resultat) råder det delade meningar bland de intervjuade om hur svårt det är att väga samman mätetal för att få en siffra på kvaliteten för ett systemutvecklingsprojekt. Det kan, på grund av ändrade förutsättningar och projekts olika utformningar och krav, diskuteras hur en viktning av dessa mätetal skulle kunna ske för att ge en status på ett systemutvecklingsprojekt.

Viktningen kan alltså skilja sig mellan olika projekt och olika tidpunkter i ett projekt. Vid en implementering av en sammanvägning är det därför viktigt att det är möjligt att anpassa vikterna i olika projekt och justera dessa under projektets gång.

I figur 2 beskrivs en möjlig översiktlig modell för hur man kan vikta samman och presentera mätetal. I modellen är projektkvalitet indelad i produktkvalitet och en kombination av tid och resurser. De två delarna presenteras med hjälp av olika diagram som beskriver t.ex. de kombinationer av mätetal som diskuterades tidigare i detta avsnitt. Utifrån dessa kombinationer räknas sedan ett värde fram för respektive del. Dessa kan sedan viktas och räknas samman till ett värde som beskriver kvaliteten eller hälsan på projektet.



Figur 2 – Vår modell över hur man kan presentera information om projektkvalitet och dess delar.

Det kan diskuteras om modellen i figur 2 är lämplig att använda, eller om en annan modell är att föredra. Det som kan argumenteras vara en fördel med denna är att man får en siffra på hur ett projekt mår som är uträknad med hänsyn till de prioriteringar som finns. Denna siffra skulle kunna presenteras t.ex. i form av en mätare eller graf som kan fungera som en varningsflagga för om problem uppstår i ett projekt. För att få insikt i vad som påverkar kvaliteten/hälsan kan man sedan studera de mätetal som beskriver de olika delarna. Eftersom viktningen kan skilja sig mellan olika projekt och olika tidpunkter, krävs det att man hela tiden anpassar viktningen efter de prioriteringar som finns i projektet. Det är dock inte helt klart vem som ska göra detta och hur detta ska göras. Det kan tänkas att man anpassar vikterna för att få en bra siffra över projektets kvalitet av olika anledningar. Kanske tror man att siffran är till för att visa hur väl utvecklingsteamet arbetar istället för hur projektet mår och därför anpassar vikterna för att ta hänsyn till förändrade förutsättningar som inte kunnat förutspås. Det är därför viktigt att man ser till att alla är medvetna om syftet med denna siffra och hur förändrade förutsättningars påverkan på projektkvalitet ska tolkas.

5.5 Vad bidrar studierna till?

Om modellen som beskrivs i avsnitt 5.4 skulle implementeras, skulle den kunna visa sig vara väldigt användbar för både projektledare och chefer. Då kan de först få en snabb överblick över flera projekts kvalitet/hälsa och sedan följa upp orsakerna till varför ett specifikt projekt har en bra eller mindre bra kvalitet/hälsa. Projektledare kan även reflektera kring vad vissa mätare eller grafer säger tillsammans med utvecklare för att se vad som skulle kunna göras för att skapa bättre produktkvalitet och hålla sig inom tidsramen för projektet.

Eftersom vi sedan tidigare inte har någon erfarenhet av TFS handlade litteraturstudien mycket om att utöka vår kunskap inom området. På grund av att vi inte har använt oss praktiskt av TFS är våra kunskaper om detta dock ändå begränsade. Detta i kombination med att de intervjuades kunskaper kring detta ämne kraftigt varierade kan ha gjort resultatet av intervjuerna begränsade. Denna variation i kunskaper kring TFS kan vara en följd av att de som arbetar på Trafikverket inte använder det till fullo, och framför allt att det används på olika sätt. Det finns t.ex. inga regler som säger hur Work Items ska användas eller hur byggen ska konfigureras. Det bör därför sättas upp någon form av standard för hur TFS ska användas. Den teoretiska bakgrunden (i synnerhet avsnitt 2.6) i denna rapport skulle kunna ligga till grund för skapandet av en sådan.

Även om resultatet kan ha påverkats negativt av de varierande kunskaperna om TFS har det dock gett bra underlag för diskussioner och kan ligga till grund för fortsatta studier.

5.6 Vidare studier

Detta ämne är relativt outforskat och i och med denna studie läggs en grund som kan användas utgångspunkt till fortsatta studier:

- Det finns möjligheter att göra en sammanvägning av mätetal, men att det kan vara svårt. Ett förslag har getts i form av en övergripande modell över hur detta skulle kunna göras. Denna modell behöver dock utvecklas vidare, implementeras och testas.
- Atkinson [3] nämner att triangeln av tid, resurser och produktkvalitet är en grundläggande modell. Det kan därför tänkas att man i vidare studier tar hänsyn till detta och eventuellt utforskar alternativ.
- Det har getts förslag på mätetal som kan användas för att ge en bild på kvalitet i systemutvecklingsprojekt. Dessa mätetal behöver dock testas för att ta reda på om de är tillräckliga för detta ändamål. Är det kanske nödvändigt att t.ex. utöka de Work Items som finns med nya attribut för att skapa nya mätetal?
- Det kan tänkas att tredjepartskomponenter skulle kunna användas för att tillsammans med TFS ge en mer komplett bild över kvaliteten på systemutvecklingsprojekt. Det är kanske möjligt att använda en sådan för att utvärdera t.ex. användbarheten med hjälp av respons från användare.

6 Slutsatser

Detta examensarbete innefattade tre frågeställningar:

- Vilka mätetal från TFS kan användas för att ge en relevant bild av kvaliteten på ett systemutvecklingsprojekt, och hur kan dessa användas till detta?
- Kan man väga samman mätetal från TFS för att få ett övergripande värde som beskriver kvaliteten på ett systemutvecklingsprojekt?
- Är TFS tillräckligt för att ge en bra bild av kvaliteten på ett systemutvecklingsprojekt

6.1 Mätetal

6.1.1 Produktkvalitet

- **Pålitlighet** kan mätas med Code coverage i kombination med antingen antal buggar eller andel tester som lyckats. Code churn kan användas för att ge en inblick i hur ett systems pålitlighet har påverkats av förändringar i koden.

- **Underhållbarhet** kan mätas med hjälp av Code metrics, vilket är ett verktyg som kan köras automatiskt i samband med byggen.

- **Effektivitet** kan mätas med hjälp av stresstester (Load tests).

- **Funktionalitet** kan mätas med kodade UI-tester (Coded UI tests) och/eller manuella tester. Hur man väljer mellan dessa beror på vilket alternativ som anses ge bäst bild av funktionalitet i förhållande till det arbete som krävs för att skapa, underhålla och utföra dessa.

- **Användbarhet** kan möjligen mätas med kodade UI-tester. Detta visar dock enbart ifall systemet gör det som utvecklaren förväntar sig att det ska göra. Man borde istället försöka få in användarens synpunkt genom t.ex. manuella användartester kombinerat med intervjuer/enkäter.

- **Portabilitet** är troligtvis svårt att mäta utifrån TFS eftersom inget direkt stöd finns för detta. Det kan hända att finns tredjepartskomponenter för att mäta detta.

6.1.2 Tid och resurser

Det kan vara svårt att utvärdera arbetsfördelningen i ett projekt utifrån TFS. Att mäta tid och resurser handlar till stor del om att ta reda på om man blir klar i tid eller inte med de resurser som finns tillgängliga. Detta kan göras med hjälp av Burndown-rapporter. Dessa i sin tur, utifrån data från Tasks, visar hur mycket av den estimerade tiden som gjorts och hur mycket som är kvar för en Sprint/Interation. Från ett utvärderingsperspektiv kan Unplanned work användas för att visa hur mycket oplanerat arbete som tillkommit under en Sprint/Iteration. Detta kan även användas som varningsflagga för om man har tagit åt sig för mycket ändringar i slutet av en Sprint/Iteration.

6.2 Sammanvägning

Det anses vara möjligt att göra en sammanvägning av mätetal för att visa kvaliteten på ett utvecklingsprojekt, men det kan vara en svår uppgift. Om de olika aspekterna i triangeln ska användas i en sammanvägning är det viktigt att tänka på att viktningen kan skilja sig mellan olika projekt. Främst i avseendet att det i vissa projekt prioriteras

att man blir klar i tid och i andra projekt prioriteras produktkvaliteten. Därför måste man i varje projekt anpassa viktningen efter de krav som finns för att resultatet ska vara relevant.

6.3 Är TFS tillräckligt?

Om man ser till de olika aspekterna i triangeln kan det diskuteras om TFS har tillräckligt stöd för att ge en bild av kvaliteten för dessa. Eftersom Burndown är en väldigt central del i tidsuppskattning kan det argumenteras för att TFS ger tillräckligt stöd för att utvärdera tidsaspekten av projektkvalitet. För resursaspekten ges visst stöd för att mäta fördelningen av resurser, men som diskuterats tidigare finns problem med detta. Detta leder till att man får förlita sig till tidsaspekten för att utvärdera resursaspekten, vilket är en begränsning då detta endast ger en väldigt översiktlig bild av denna. När det gäller produktkvalitet kan pålitlighet och underhållbarhet kan mätas relativt enkelt. Användbarhet, funktionalitet och effektivitet kan mätas om man lägger ner visst arbete på att skapa tester som mäter dessa. Portabilitet däremot, kan vara svårt att mäta utifrån TFS.

Är det då möjligt att utifrån detta ge en bra bild av kvaliteten på systemutvecklingsprojekt? Eftersom tidsaspekten går att mäta, produktkvalitet kan mätas till en ganska hög grad och resursaspekten kan uppskattas utifrån tidsaspekten kan man argumentera för att en någorlunda bra bild skulle kunna ges. Detta kräver att man gör lite merarbete, i form av att t.ex. skriva olika typer av tester, för att göra det möjligt att mäta de flesta av kvalitetsattributen i produktkvaliteten. Görs inte detta får man troligtvis endast en väldigt begränsad bild av projektkvalitet som missar flera viktiga delar.

För att det ska vara möjligt att få en bra bild av projektkvalitet från TFS krävs det att de projekt som ska mätas använder det i tillräcklig utsträckning. Detta kan uppnås genom att man sätter upp en standard för vilka delar av TFS som ska användas och hur de ska användas.

Litteraturförteckning

[1] M Woodward, G Holliday, B Keller E Blankenship, *Profesional Team Foundation Server 2010*, 1st ed. Indianapolis, USA: Wiley Publishing, Inc., 2011.

[2] Microsoft. Every project is a triangle. [Online]. http://office.microsoft.com/en-us/project-help/every-project-plan-is-a-triangle-HA001021180.aspx (besökt 31/5)

[3] R Atkinson, "Project management: cost, time and quality, two best guesses and a phenomen, its time to accept other success critera," *Internation Journal of Project Management*, vol. 17, no. 6, pp. 337-342, 1999.

[4] J Ronkainen, J Warsta P Abrahamsson O Salo, *Agile software development methods - Review and analysis*, 1st ed. Oulu, Finlad: VTT Technical Research Centre of Finland, 2002.

[5] M Azuma N Bevan, "Quality in use: incorporating human factors into the software engineering lifecycle," in *Software Engineering Standards Symposium and Forum*, Teddington, 1997, p. 169.

[6] D.J Ferrand J.P Paquin J Couillard, "Assessing and controlling the quality of a project end product: the earned quality method," *Engineering Management IEEE*, vol. 47, no. 1, p. 88, Februari 2000.

[7] R.G. Dromey, "A model for software product quality," *Software Engineering, IEEE*, vol. 21, no. 2, p. 146, Februari 1995.

[8] S.G. Kim, C.S. Chung H.W. Jung, "Measuring software product quality: a survey of ISO/IEC 9126," *Software, IEEE*, vol. 21, no. 5, p. 88, September-Oktober 2004.

[9] P Borque, A Abran, C Laporte W Suryn, "Software product quality practices - quality measurement and evaluation using TL9000 and ISO/IEC 9126," in *Software Technology and Engineering Practice*, Montreal, Oktober 2002, p. 156.

[10] N Ehsan, E Mirza, S.Z Sarwar A Ahmed S Ahmad, "Agile software development: Impact on productivity and quality," in *Management of Innovation and Technology*, Islamabad, 2010, p. 287.

[11] Agile Sweden. [Online]. http://www.agilesweden.org/ (besökt 27/04-2011)

[12] Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas Kent Beck Mike Beedle. (2001) Manifest för Agil systemutveckling. [Online]. http://agilemanifesto.org/iso/sv/ (besökt 27/04-2011)

[13] S. Lin, S.J. Simon A.I. Concepcion, "The RMT (Recursive Multi-Threaded) tool: a computer aided software engineering tool for monitoring and predicting software development progress," in *Software Engineering*, San Bernardino, 1999, p. 660.

[14] O Hazzan, Y Dubinsky D Talby A Keren, "Agile software testing in a large-scale project," *Software, IEEE*, vol. 23, no. 4, p. 30, Juli-Augusti 2006.

[15] M Puleio, "How Not to do Agile Testing," in *Agile Conference*, Redmond, 2006, p. 7.

[16] F Deissenboeck, E Juergens, B Hummel, B, Mas y Parareda, M Pizka S Wagner, "Tool Support for Continous Quality Controll," *Software, IEEE*, vol. 25, no. 5, p. 60, September-Oktober 2008.

[17] B Keller, A Krishnamoorthy, Martin Woodward M Gousset, *Professional Application Lifecycle Management with Visual Studio 2010*, 1st ed. Indianapolis, USA: Wiley Publishing, Inc., 2010.

[18] Team Foundation Server Fundamentals: A Look at the Capabilities and Architecture. [Online]. http://msdn.microsoft.com/en-us/library/ms364062.aspx (besökt 27/04-2011)

[19] User Story (Agile). [Online]. http://msdn.microsoft.com/sv-se/library/dd380634.aspx (besökt 27/04-2011)

[20] Test Case (Agile). [Online]. http://msdn.microsoft.com/en-us/library/dd380712.aspx (besökt 27/04-2011)

[21] Shared Steps (Agile). [Online]. http://msdn.microsoft.com/en-us/library/dd728086.aspx (besökt 27/04-2011)

[22] Task (Agile). [Online]. http://msdn.microsoft.com/en-us/library/dd380700.aspx (besökt 27/04-2011)

[23] Bug (Agile). [Online]. http://msdn.microsoft.com/en-us/library/dd380645.aspx (besökt 27/04-2011)

[24] Visual Studio Scrum 1.0. [Online]. http://msdn.microsoft.com/en-us/library/ff731587.aspx (besökt 27/04-2011)

[25] Product Backlog Item (Scrum). [Online]. http://msdn.microsoft.com/en-us/library/ff731576.aspx (besökt 27/04-2011)

[26] Task (Scrum). [Online]. http://msdn.microsoft.com/en-us/library/ff731574.aspx (besökt 27/04-2011)

[27] Test Case (Scrum). [Online]. http://msdn.microsoft.com/en-us/library/ff731578.aspx (besökt 27/04-2011)

[28] Shared Steps (Scrum). [Online]. http://msdn.microsoft.com/en-us/library/ff731581.aspx (besökt 27/04-2011)

[29] Bug (Scrum). [Online]. http://msdn.microsoft.com/en-us/library/ff731589.aspx (besökt 27/04-2011)

[30] Impediment (Scrum). [Online]. http://msdn.microsoft.com/en-us/library/ff731580.aspx (besökt 27/04-2011)

[31] Sprint (Scrum). [Online]. http://msdn.microsoft.com/en-us/library/ff731582.aspx (besökt 27/04-2011)

[32] Testing the Application. [Online]. http://msdn.microsoft.com/en-us/library/ms182409.aspx (besökt 2/5-2011)

[33] Code Metrics Values. [Online]. http://msdn.microsoft.com/en-us/library/bb385914.aspx (besökt 20/5-2011)

[34] Integrating Code Metrics in TFS 2010 Build. [Online]. http://geekswithblogs.net/jakob/archive/2011/01/30/integrating-code-metrics-in-tfs-2010-build.aspx (besökt 20/5-2011)

[35] Components of the Data Warehouse for Team Foundation. [Online]. http://msdn.microsoft.com/en-us/library/ms244687.aspx (besökt 27/04-2011)

[36] Creating and Customizing TFS Reports. [Online]. http://msdn.microsoft.com/en-us/library/ff647430.aspx (besökt 27/04-2011)

[37] Reports (Agile). [Online]. http://msdn.microsoft.com/en-us/library/dd380714.aspx (besökt 27/04-2011)

[38] Bug Status Report. [Online]. http://msdn.microsoft.com/en-us/library/dd380736.aspx (besökt 29/04-2011)

[39] Bug Trends Report. [Online]. http://msdn.microsoft.com/en-us/library/dd380674.aspx (besökt 29/04-2011)

[40] Reactivations Report. [Online]. http://msdn.microsoft.com/en-us/library/dd380731.aspx (besökt 29/04-2011)

[41] Build Quality Indicators Report. [Online]. http://msdn.microsoft.com/en-us/library/dd380683.aspx (besökt 29/04-2011)

[42] Build Success Over Time Report. [Online]. http://msdn.microsoft.com/en-us/library/dd380643.aspx (besökt 29/04-2011)

[43] Build Summary Report. [Online]. http://msdn.microsoft.com/en-us/library/dd380708.aspx (besökt 29/04-2011)

[44] Burndown and Burnrate Report (Agile). [Online]. http://msdn.microsoft.com/en-us/library/dd380678.aspx (besökt 29/04-2011)

[45] Remaining Work Report. [Online]. http://msdn.microsoft.com/en-us/library/dd380673.aspx (besökt 29/04-2011)

[46] Status on All Iterations Report. [Online]. http://msdn.microsoft.com/en-us/library/dd380706.aspx (besökt 29/04-2011)

[47] Stories Overview Report (Agile). [Online]. http://msdn.microsoft.com/en-us/library/dd380648.aspx (besökt 29/04-2011)

[48] Reporting on Testing Progress for Test Plans. [Online]. http://msdn.microsoft.com/en-us/library/dd286682.aspx (besökt 29/04-2011)

[49] Stories Progress Report (Agile). [Online]. http://msdn.microsoft.com/en-us/library/dd380641.aspx (besökt 29/04-2011)

[50] Unplanned Work. [Online]. http://msdn.microsoft.com/en-us/library/ee707132.aspx

[51] Test Case Readiness Report. [Online]. http://msdn.microsoft.com/en-us/library/dd380713.aspx (besökt 29/04-2011)

[52] Test Plan Progress Report. [Online]. http://msdn.microsoft.com/en-us/library/dd380702.aspx (besökt 29/04-2011)

[53] Release Burndown (Scrum). [Online]. http://msdn.microsoft.com/en-us/library/ff731579.aspx (besökt 29/04-2011)

[54] Sprint Burndown (Scrum). [Online]. http://msdn.microsoft.com/en-us/library/ff731588.aspx (besökt 29/04-2011)

[55] Velocity (Scrum). [Online]. http://msdn.microsoft.com/en-us/library/ff731575.aspx (besökt 27/04-2011)

[56] Dashboards (Agile). [Online]. http://msdn.microsoft.com/en-us/library/dd380719.aspx (besökt 27/04-2011)

[57] G Hughes N Murray, *Writing up your University Assignments and Research Projects - A practical handbook*, 1st ed. Berkshire, England: McGraw Hill, 2008.

Bilaga 1 – Intervjufrågor

1. Tror du att modellen med tid, resurser och produktkvalitet är lämplig att använda för att beskriva kvaliteten på ett systemutvecklingsprojekt? Om inte, vad ska man använda för modell?

2. Tror du det är möjligt att väga samman produktkvalitet, tid och resurser för att ge en helhetssyn av kvaliteten på ett utvecklingsprojekt?
3. Hur tror du man ska vikta de olika mätetalen?

4. Vad för slags data/information tror du att man behöver för att kunna utvärdera kvaliteten på en produkt?
5. Vilka mätetal tror du man kan använda för att utvärdera detta?
6. Tror du att dessa är tillräckliga?
7. Hur tror du man kan använda dem på bästa sätt?

8. Vad för slags data/information tror du att man behöver för att kunna utvärdera hur ett projekt ligger till i tidsplanen?
9. Vilka mätetal tror du man kan använda för att utvärdera detta?
10. Tror du att dessa är tillräckliga?
11. Hur tror du man kan använda dem på bästa sätt?

12. Vad för slags data/information tror du att man behöver för att kunna utvärdera om ett projekt har lagom med arbetskraft och om arbetskraften är bra fördelad i projektet?
13. Vilka mätetal tror du man kan använda för att utvärdera detta?
14. Tror du att dessa är tillräckliga?
15. Hur tror du man kan använda dem på bästa sätt?

Bilaga 2 – Mätetal

Buggar

- Antal aktiva buggar
- Antal lösta buggar
- Antal stängda buggar
- Aktiva buggar per prioritet eller allvarlighet
- Antal reaktiverade buggar
- Hastigheten som buggar öppnas
- Hastigheten som buggar stängs
- Hastigheten som buggar blir lösta
- Antal aktiva buggar som är tilldelade olika personer
- Antal lösta buggar som är tilldelade olika personer

Byggen

- Antal rader kod som förändrats i ett bygge
- Andel kod som är täckt av tester för ett bygge
- Antal tester som lyckats för ett bygge
- Antal tester som misslyckats för ett bygge
- Antal tester som är resultatlösa för ett bygge
- Resultat av byggen
- Andel lyckade tester för ett bygge

Work Items

- Antal aktiva Work Items (Task, Bug, User Story)
- Antal lösta Work Items (Task, Bug, User Story)
- Antal stängda Work Items (Task, Bug, User Story)

Iterationer

- Antal User Stories stängda för en iteration
- Antal aktiva buggar för en iteration
- Antal lösta buggar för en iteration
- Antal stängda buggar för en iteration
- Antal Work Items som lades till i planeringen av en iteration
- Antal Work Items som lades till under en iteration

User Story och Task

- Andel lyckade test för den senaste testkörningen av en User Story
- Andel misslyckade test för den senaste testkörningen av en User Story
- Andel test ej körda för den senaste testkörningen av en User Story
- Antal aktiva buggar för en User Story
- Antal lösta buggar för en User Story
- Antal testkonfigurationer för en User Story
- Timmar kvar för en User Story
- Andel arbetad tid för en User Story
- Andel återstående tid för en User Story
- Andel arbetad tid som gjort nyligen för en User Story
- Mängd tilldelat arbete per person (timmar utifrån Tasks)

Test Cases

- Antal Test Cases som är i designläge
- Antal Test Cases som är färdiga
- Antal lyckade tester för en testkörning
- Antal misslyckade tester för en testkörning
- Antal blockerade tester för en testkörning
- Antal tester som inte körts för en testkörning

Iterationer/Sprints

- Antal arbetade timmar för en iteration
- Antal återstående timmar för en iteration
- Antal arbetade per dag för en iteration
- Originalestimering av tid för en iteration
- Arbetad tid som gjorts för en iteration
- Återstående tid för en iteration
- Återstående arbete i början av en Sprint
- Antal återstående timmar som håller på att utföras för en sprint