CHALMERS

Westimate, ett projektverktyg för att förbättra tidsuppfattning och estimering

Master of Science Thesis in Computer Science and Engineering

CHRISTIAN BERTILSSON

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Göteborg, Sweden, June 2009

Westimate, ett projektverktyg för att förbättra tidsuppfattning och estimering

Christian Bertilsson

Examiner: Christer Carlsson

Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden, June 2009

Sammanfattning

Uppdragsgivaren till detta examensarbete, Westbahr AB, var i behov av ett system för att lagra, följa och dokumentera de arbetsuppgifter som utförs i projekt på företaget. De upplevde ständigt återkommande problem med att uppskatta, kontrollera och hålla sig inom projekts tidsramar. De ville därför även implementera ett stöd, i systemet, för att uppmuntra användarna att utveckla sin förmåga att estimera uppgifter och omfattningen av projekt.

Ett antal metoder att planlägga och se på strukturen i projekt har undersökts för att kunna räkna på när ett projekt kan slutföras. Klassiska scheman, som Gantt och PERT, har traditionellt använts. De förlitar sig dock på estimat som anges utan kritik, som återkoppling med historiska data, och är svåra att handskas med när projekt förändras. Som ofta är fallet för Westbahr AB.

En webbapplikation, som kallas Westimate, har genom detta arbete designats och börjat utvecklas. I Westimate har en förenklad variant av en algoritm föreslagen av Joel Spolsky, Evidence Based Scheduling, implementerats. Algoritmen försöker förutsäga hur mycket längre tid en samling uppgifter som har estimerats egentligen kommer att ta att utföra. Detta görs genom en Monte Carlo-simulering som anpassar estimaten efter tidigare resultat från historiska data. Algoritmen ger en diskret funktion av approximativa sannolikheter som täcker ett öppet intervall - från simuleringens mest optimistiska beräkning med låg sannolikhet, till den mest pessimistiska med hög sannolikhet.

Det framtagna systemet uppfyller majoriteten av de krav som ställts. Systemet har tagits i drift av Westbahr AB och används även i mindre utsträckning av ett par närstående företag. Många utvidgningar och förbättringar finns kvar att göra, men det är helt i sin ordning och går väl ihop med den adaptiva utvecklingsmetod som använts.

Algoritmen behöver utvärderas i avseende på hur bra stöd den ger. För detta krävs ett större underlag av historiska data och resultat från flera genomförda projekt. Algoritmen kommer att justeras och utvecklas vidare vid behov efter en utvärdering.

Nyckelord; uppgiftshantering, projektstöd, estimering, Evidence Based Scheduling, Coldfusion, Farcry CMS, Monte Carlo-simulering.

Westimate, a project tool for enhanced time perception and estimation

Abstract

The client of this thesis, Westbahr AB, was in need of a system to store, monitor and document the tasks performed in projects at the company. They experienced recurring problems in estimating, controlling and keeping project deadlines. Therefore, they wanted to implement a supporting feature in the system to encourage users to develop their ability to estimate tasks and scope of projects.

A number of methods to plan and look at the structure of projects, to be able to calculate when a project can be completed, have been studied. Classical charts, as Gantt and PERT, have traditionally been used. They however rely on estimates used without criticism as feedback in historical data, and these diagrams are difficult to deal with when projects change. As is often the case for Westbahr AB.

A web application, called Westimate, has through this work been designed and begun to be developed. A simplified variant of an algorithm proposed by Joel Spolsky, Evidence Based Scheduling, has been implemented in Westimate. The algorithm tries to predict how much longer a collection of tasks that has been estimated will take to perform. This is done through a Monte Carlo simulation which adjusts estimates against earlier results from historical data. The algorithm provides a discrete function of approximate probabilities that covers an open range - from the simulations most optimistic estimate of low probability, to the most pessimistic with a high probability.

The developed system meets the majority of the requirements set. The system has been taken into operation by Westbahr AB and is also used to a lesser extent by a couple of related companies. Many extensions and improvements are left to do, but that was expected and goes well with the adaptive development method used.

The algorithm needs to be evaluated in terms of how well it provides support. This requires a larger base of historical data and results from several completed projects. The algorithm will be adjusted and further developed if necessary after an evaluation.

Keywords; task management, project support, estimation, Evidence Based Scheduling, ColdFusion, Farcry CMS, Monte Carlo simulation.

Förord

Det här arbetet är mitt avslutande moment på civilingenjörsprogrammet Informationsteknik vid Chalmers Tekniska Högskola. Arbetet har utförts vid Institutionen för data- och informationsteknik (D&IT) med inriktning datakommunikation.

Ett speciellt tack riktas till min handledare på Westbahr AB, Peter Thorin, för initiativet till detta projekt. Stora tacksägelser går även ut till framförallt Mattias Bergsten och Daniel Niklasson på företaget för intressanta diskussioner, idéer och stöd.

Ett tack riktas även till studiekamraterna, ingen nämnd ingen glömd, som har granskat arbetet och givit synpunkter. Det största tacket går slutligen till min examinator på Chalmers, Christer Carlsson, för ovärdelig vägledning vid slutförandet av denna rapport.

Göteborg, Juni 2009

Christian Bertilsson

Innehållsförteckning

1 Inledning

1.1 Bakgrund

1.1.1 Westbahr AB

Westbahr AB är ett ungt företag som finns i Göteborg, Stockholm, Malmö och Växjö. Från att ha verkat i mindre skala i några år, med ca 4-5 personer inblandade vid sidan av andra uppdrag, har företaget nu över 25 anställda och expanderar i snabb takt. Expansionen inleddes under sommaren 2007 då det första kontoret i Göteborg startades upp.

Företaget är i grunden ett utvecklingsbolag som tar fram anpassade och innovativa lösningar inom flera delar av IT-segmentet. Grundtanken är att genom konsulter hjälpa små och medelstora företag att effektivisera sin verksamhet. För detta erbjuds tjänster som inköpsplanering, konfigurering och drift av IT-miljöer samt utveckling av applikationer inom flera skilda områden, allt från affärssystem till underhållningsmedia.

1.1.2 Westimate

I detta examensarbete har ett projekthanteringssystem som kallas Westimate tagits fram. Namnet Westimate används framöver i rapporten om den nuvarande implementationen. I resterande delen av inledningen då "verktyget" eller "systemet" omnämns kan det lika gärna syfta på det tänkta resultatet innan arbetets påbörjan.

1.2 Problembeskrivning

Westbahr AB var i behov av ett system där arbetsuppgifterna som utförs på företaget registreras och hanteras. Ett uppgiftscentrerat projekthanteringssystem.

Kravspecifikationen från Westbahr AB nämner bland annat att varje uppgift ska placeras i ett projekt. Uppgifterna ska fördelas bland systemets användare, resurser, som vanligtvis är företagets anställda konsulter. När uppgifterna utförs fyller resurserna på med information om uppgiften i sig och om hur arbetet med uppgiften fortskrider. Eventuell internkommunikation och relaterade e-postkonversationer ska ske i anslutning till den aktuella uppgiften i systemet.

Företaget har sedan starten uppmärksammat svårigheterna med att estimera och följa upp tidsåtgång i arbetet. De upplevde att det behövdes ett hjälpmedel för att kunna förbättra tidsmedvetenheten på företaget - de anställdas känsla för hur förväntad och upplevd arbetstid motsvarar den faktiska åtgången. Det var även ett problem att uppgifter föll mellan stolarna och att mycket debiterbar tid förgåtts på grund av att tidsåtgången ej registrerats och följts upp löpande i närhet till de uppgifter som utförs.

Westbahr ville med systemet undersöka möjligheterna att angripa ovan nämnda problem. Det generella problemet är med andra ord designen och implementationen av

systemet. Det specifika problemet är framtagandet av det särskilda stöd, som blir ett förslag på hur systemet kan handskas med estimeringar för att hjälpa användaren i sitt arbete med osäkra tidsramar.

Problemet att uppskatta tidsåtgången i praktiska projekt inom IT är allmänt känt och det kommer aldrig finnas några magiska lösningar [1] [2] [5]. Såvida ett projekt inte är väldigt begränsat och nya krav eller sena förändringar inte tillåts. Mjukvaruutveckling och drift av IT-miljöer sker sällan isolerat. Inte bara tekniken och insikter i det aktuella projektet förändras under processen. Kravställare som kunder och ofta även kundernas kunder samt parallella projekt bidrar alla till att tidsåtgången blir omöjlig att förutsäga.

Uppföljning av tid efteråt är i sig inget problem. Men ska uppföljningen innefatta mer än bara en enkel jämförelse av uppskattad och förbrukad tid ligger utmaningen i datainsamlingen och vilken statistik som kan tas fram ur den.

1.3 Syfte

Syftet med detta examensarbete är att påbörja utvecklingen av ett webbaserat projekthanteringsystem. I detta ingår att undersöka hur systemet ska utformas för att det ska gå att införa en lämplig metod för tidsestimeringsstöd på projektnivå. Syftet är också att resonera kring några av de metoder som finns idag och välja en som implementeras.

Arbetet utförs med förhoppningen om att det för Westbahr AB på sikt ska leda till bättre tidsramar, ökad effektivitet, större kundförtroende och en trygghet i att se hur kunderna debiteras för tiden som faktiskt går åt.

Genom att arbeta aktivt med estimat borde chansen finnas för systemets användare att utveckla förmågan att estimera och planera uppgifter och projekt, samt att analysera den egna tidsuppfattningen. Visar det sig att många av de uppgifter som estimeras tenderar att ha en stor osäkerhet i sig, även med erfarenhet av att estimera, bör en konsekvens ändå bli att användarna utvecklas i att isolera de mest osäkra delmomenten. För att allt bättre kunna bryta ut och öka kontrollen i det resterande arbetet.

De interna rutinerna på Westbahr AB ska avsevärt ha förbättrats några månader efter att systemet har satts i drift. De anställda ska uppleva att de fått en ökad överblick på vad som sker för stunden och hur projekt fortlöper. Westimate ska användas som ett naturligt stödjande inslag i allt arbete på företaget.

1.4 Avgränsningar

Huvuduppgiften i detta arbete är att analysera problemdomänen och designa systemet därefter, så att förutsättningarna för att undersöka den specifika problemställningen finns.

På grund av examensarbetets begränsade omfattning fanns det inte möjlighet att föreslå och jämföra flera alternativa metoder för att angripa det specifika problemet. Resultatet blir ett förslag på en lösning. Sedan kan arbetet tas vidare genom att vidareutveckla lösningen eller utarbeta en annan med den som utgångspunkt.

En stor del av systemets mer generella funktionalitet är inte av så stor teoretisk betydelse och har heller inget med det specifika problemet att göra. Det skulle krävas en

för stor insats att redogöra för hela systemet i detalj. Vad gäller mer detaljerade begränsningar i utformningen av systemets funktionalitet samt krav på integrationer och framtida mål, se kravspecifikation i appendix.

Systemet som tas fram kommer att vara fokuserat kring uppgiftshantering. Arbetet behandlar inte relaterade problem, som styrning av andra resurser än anställda, riskhantering eller schemaläggning med omräkning av arbetstimmar till kalendertid. Stöd för projektekonomi - det vill säga beräkningar med intäkter, omkostnader och vinst - är ett område som har diskuterats i anslutning till projektet men som ännu bara har berörts lätt i designen av systemet. Det är ett för stort område för att kunna behandlas i betydande grad i detta arbete.

Det hade varit önskvärt att genomföra en uppföljningsstudie, med ett fullskaligt praktiskt användningsfall och en utvärdering av dess resultat för att kunna uppskatta kvalitén av systemet och det specifika stödet. Men på grund av den tidsmässiga omfattningen var detta inte möjligt att genomföra.

2 Metod

Avsikten med detta avsnitt är att ge en bakgrund till hur arbetet har utförts och hur det kommande analysavsnittet av rapporten är utformat.

Eftersom arbetet har en såpass fri utgångspunkt är det en stor del av uppgiften att förstå problemet och dess domän. Att resonera kring vilka möjligheter som finns idag och kombinera väl valda koncept till en anpassad lösning.

En tidig kravspecifikation, som beskrev systemets allmäna krav, dess tänkta innehåll och mål, erhölls från Westbahr AB vid starten av projektet. Detta dokument har sedan utvecklats vidare kontinuerligt genom arbetet.

Inför arbetet påbörjades det en förstudie som en del av mitt tidigare deltidsarbete på Westbahr AB. Tanken var att närma sig problemet genom att granska projektens karaktär på företaget, identifiera behov och beskriva användarna av det kommande systemet. Detta gjordes genom en kvalitativ undersökning med diskussioner och observationer.

Denna förstudie avslutades aldrig under den perioden utan pågick en bra bit in under själva examensarbetet. Studien utökades kontinuerligt med kompletterande undersökningar. I en perfekt värld hade en separat förstudie legat till grund för analysen. I verkligheten som detta arbete har utförts i finns ingen tydlig åtskillnad i hur förstudierna samverkat med resonemang som förts under arbetets gång. Det finns därför inte något naturligt sätt att bryta ut den ur analysavsnittet.

Ur analysen växte de slutsatser fram som resultatet - designen och implementationen av systemet - har baserats på. Samt teorin som beskriver den valda tekniken för att erbjuda en lösning på den specifika problemställningen.

2.1 Utvecklingsmetod

Som utvecklingsmetod har ett agilt synsätt använts, som närmast kan beskrivas som adaptiv utveckling - ASD, Adaptive Software Development [6] - med en ständigt pågående diskussion med användare och intressenter i olika roller på företaget.

Den adaptiva utvecklingsmetodologin ansågs vara bäst lämpad under denna fas, den tidiga utvecklingen av systemet samtidigt som examensarbetet utförs. Av anledningarna att examensarbetet är utformat som ett enmansprojekt och projektets experimentella natur gällande vad som ska hända sedan. Det är möjligt att en stor del av koden som skrivs nu kan leva vidare i systemet längre fram. Arbetet ansågs vara i för liten skala och med för få inblandade för att korrekt följa en mer rigorös agil projektmetod. Förväntningen på att kunna anpassa Westimate i både stort och smått för att möta nya krav låg också till grund för detta val.

Av egna erfarenheter är det uppenbart att ett nytt system måste visas upp i sin tänkta miljö och testanvändas skarpt så fort det är möjligt. Det är då de mest användbara kraven och idéerna kommer. Highsmith [6] skriver att "Blindly following a plan produces the product you intended, just not the product you need. It is actually deviations to the plan that will guide toward the correct solution". Viktiga aspekter av kärnproblemen framträder och användarens åsikter kan tas med in i lösningarna. När systemet är i

användning och funktionalitet tillkommer efter hand är det enklare att få med sig användare och andra intressenter i designprocessen. Förändringar i systemet bör då följa deras verklighetsbaserade förväntningar och inte innebära förvirrande överraskningar. En produkts användbarhet visar sig i samspelet mellan produkten och dess användare över en tidsperiod, i användning [8].

I diskussionerna på Westbahr om behoven för Westimate hänvisades det ofta till särskilda detaljer som lagts märke till och uppskattats i andra applikationer. Att inspireras kan vara bra, om föremålet för inspiration har används med framgång och redan är ett känt koncept främjar det inte användarvänligheten att ta fram en annan [8]. I detta projektet har dock många attribut i gränssnittet tagits fram ganska fritt utan att följa en särskild metod, med motiveringen att det inte har prioriterats på grund av tidsbrist och att det funnits en vilja att experimentera.

2.2 Implementeringsprocess

Arbetet har utförts till och från under en tidsperiod på 1,5 år. Andra projekt har utförs parallellt, under vissa perioder har det enbart varit annat arbete och ibland har fokus helt varit på Westimate. Kringliggande funktionalitet och kodning av användargränssnitt har blivit en stor del av den totala insatsen i hela projektet.

Det har varit ett medvetet val att låta processen ta sin tid, med förhoppningen om att få ett bra resultat gällandes systemet i sig. Istället för att hasta fram en dåligt genomarbetad plattform som bara lever upp till examensarbetets krav men blir svår att utveckla vidare.

Detta val har inte bara påverkat utvecklingen av systemet utan även för processen med examensarbetet har det spelat in. Ett stort antagande som gjordes tidigt var att den specifika lösningen skulle börja implementeras när det fanns testdata från ett verkligt projekt tillgängligt. Det förväntades att det då skulle dyka upp flera viktiga avvikelser och specialfall som behövde hanteras. Detta har stoppat upp processen åtskilliga gånger eftersom mycket annat grundläggande har behövt uppfyllas i systemet innan data kunnat samlats in skarpt.

Förloppet för implementeringen av Westimate kan delas upp i tre iterativa steg.

Steg 1: Prototyp för att hitta grundstrukturen

En prototyp togs fram för att strukturera upp relationer och inmatning av testdata. Vilket gav en grund för att laborera med de mest grundläggande objekten, vrida och vända på dem och hoppas på att det som framkommer har en bra inkapslad utvecklingsbarhet och hög hållbarhet i relationen till andra grundläggande objekt.

Till att börja med var prototypen mer utav ett skal i form av en komprimerad panel med formulärrutor. Varje formulär kunde skapa objekt eller relationer. Till detta skrevs testvyer som plockade ut data.

Panelen bestod till sist av ett tjugotal boxar som itererades igenom 3 gånger med många justeringar som påverkade andra rutor. Ett domändiagram togs fram för strukturen som erhölls.

Steg 2: Användbar prototyp som införs i verksamheten

Prototypen utvecklades vidare till ett enkelt system som kunde börja användas på riktigt. Systemet infördes först i verksamheten i mindre skala. Enbart ett projekt, det

som behandlar allmän e-postsupport, sattes igång. För att fånga upp barnsjukdomar innan systemet skulle användas i större skala.

Steg 3: Insamling av verkliga krav och nya problem, som löses efter hand

Den tredje och sista fasen är den större delen av processen, det arbete som pågår kontinuerligt. Det är nu mer avancerad funktionalitet och fler plötsliga förändringar införs. Och insikter från verkligheten hinner ikapp systemet. Den adaptiva utvecklingsmodellens tre delmoment - speculate, collaborate, learn [6] - används för att studera och utveckla Westimate.

Nya funktioner och ändringar spekuleras fram, införs och testas i användning. Om ändringarna ger upphov till några problem eller nya lärdomar får de diskuteras och laboreras vidare med i en ny spekulation om vilka konsekvenser åtgärdandet av problemet kan innebära. Dessa steg repeteras sedan och utvecklingen drivs framåt.

Det har antagits att potentiella oklarheter som inte är väl dokumenterade eller avgränsade inte bör få ligga kvar i kod som används utan de ska genast angripas och redas ut eller kapslas undan inför framtiden. Detta direktiv kom ur egna erfarenheter av andra adaptivt växande projekt.

Oftast har en lista av uppgifter sammanställts till en så kallad sprint [19] med saker att utföra under en koncentrerad period. Genom ett möte med nyckelpersoner på företaget har uppgifterna prioriterats. De bildar då en agenda med de närmsta delmålen, vilket ger en hanterlig arbetsmängd att fokusera på åt gången.

3 Analys

Denna del ger i första hand en fördjupning i bakgrunden till problemställningen. Kapitlets upplägg har en del gemensamt med hur en förstudie skulle ha presenterats, framförallt det första avsnittet.

Den tekniska plattformen för systemets arkitektur framgick av kravspecifikationen och har därmed inte valts, men de produkter, ramverk och tekniker som systemet har implementerats i presenteras kort i detta kapitel.

Sammanfattningsvis är det följande frågor som belyses:

- Vad för projekt utförs på Westbahr och hur är de upplagda? Finns det särskilda skillnader i IT-projekt jämfört med andra tekniska områden med längre tradition? Vilka aspekter finns i att hantera arbete i projekt och vad bör beaktas vid utveckling av ett system som Westimate till en organisation som Westbahr? Hur ska systemet införas hos Westbahr så att det kommer i användning?

- Vilka typer av arbetsuppgifter och projekt är relevanta för systemet? Hur kan de estimeras? Vad finns det för möjligheter att genom ett system erbjuda stöd för ökad förståelse av hur tiden används, öka de anställdas tidsuppsfattning och förmåga att på sikt uppskatta uppgifter och projekt bättre?

Observera att detta inte har utförts som en separat förstudie innan utvecklingen av systemet påbörjades. Flera av resonemangen i analysen har förts fritt under arbetets gång och slutsatserna som sammanställts i kapitlets avslutande avsnitt har tillkommit efter hand.

De slutsatser och lösningar som presenteras har legat till grund för resultatet och det resterande arbetet som har genomförts. Det generella problemet att designa systemet i sin helhet baseras till stor del på kravspecifikationen, men resonemang och slutsatser i analysen har också påverkat. Den teoretiska referensramen som problemdomänen främst har utgått ifrån är avsnitten om projekt, estimat och metoder för projektestimering. De är inriktade på den specifika problemställningen - systemets mål att kunna stödja användarens användning av tidsuppskattningar.

3.1 Anknytning till verkligheten

För att bemöta problemet behöver först sammanhanget som problemställningen uppstod i beskrivas - den miljö som verktyget Westimate ska verka i.

3.1.1 Identifiering av behov och systemets användare

Det första steget i framtagandet av Westimate var att identifiera behovet. Det var tydligt att ett projekthanteringssystem behövdes införas och tanken på denna uppgift som ett examensarbete fanns över ett halvår innan uppgiften påbörjades.

En kvalitativ undersökning för att observera företagets rutiner och behov påbörjades i liten skala redan då. Det diskuterades med företagsledningen om vilka grundläggande

funktionella krav som skulle ställas, för att bilda en grundläggande uppfattning om problemdomänen.

Denna metod har hela tiden varit informell och bestått av observationer och samtal med nyckelpersoner med skilda arbetsuppgifter. Möten och intervjuer har planerats in när det har funnits en god anledning att reda ut något eller när det har funnits tid över.

Undersökningen har förändrats i takt med företagets tillväxt. Fler roller och åsikter har anslutit sig och gjort att önskemålen har behövt struktureras. Beskrivningar av genomtänkta idéer och många lösryckta förslag har samlats in och grupperats ihop godtyckligt.

Karaktären på diskussionerna förändrades över tid, från att först ha handlat om att ge mig en insikt i delar av verksamheten och de andras syn på vad som behövs, till att handla om att gemensamt resonera kring de bästa lösningarna på relevanta och bra förslag som användarna själva kom med.

Denna metod avsåg även att täcka in all behövlig analys över de vanligaste användarna av systemet. Sedan tillkommer så kallade externa resurser, gästanvändare, vanligtvis i form av kunder. Dessa har inte analyserats mer än i ett kort resonemang om hur vanligt det kan bli att de får tillgång till systemet, vilka olika bakgrunder de kan ha och hur mycket de behöver känna till för att använda de delar som är relevanta för dem. Det förutsätts att de får hjälp med detta, ambitionen har inte varit att anpassa gränssnittet för dem inledningsvis.

3.1.2 Westbahrs tidiga år

Uppdragsgivaren för detta examensarbete, Westbahr AB, är en ung organisation med växtvärk. Grundarna består av ett par entreprenörer som besitter mycket stor teknisk kompetens, driv och praktisk erfarenhet inom systemutveckling mot internetapplikationer samt uppstrukturering och drift av komplexa IT-miljöer. Men struktur och ordning i det egna arbetet har upplevts som ett allt större problem - med många spruckna tidsramar som följd.

Från starten har det på sätt och vis saknats en ordningshållande kultur på företaget. Westbahrs första år har präglats av ett ad hoc-arbetssätt. Förmodligen på grund av god vilja inför att kunden alltid vill ha det kommande arbetet gjort helst igår - och det har hört till vanligheterna att det kunden ber om löses med ingrepp utan att spekulera ett extra varv kring konsekvenserna. Många mindre företag vill uttryckligen inte betala för mer än vad som anses behövligt för stunden.

Sedan återkommer kunden med nya direktiv som löses efter hand med de tidigare ingreppen som förutsättning. Är kunden ovetande om vad som ska göras i nästkommande steg och inte tänker igenom vad de egentligen vill uppnå uppstår det ofta extra oförutsett besvär. Att tänka framåt är heller inte någon bra garanti, oväntade förändringar har snarare varit regel än undantag. Många situationer bedöms till och med på förhand som så föränderliga att långsiktig planering inte gör någon större nytta. Men det blir till stor del kundens ansvar och svårt att kontrollera då uppenbart ogenomtänkta krav alltid ifrågasätts och diskuteras först.

Ett par kunder har funnits med sedan början och det kan nog ha blivit ett problem. Mindre och medelstora företag, ofta där det har funnits en god personlig kundkontakt. Det blir svårt att införa rutiner när saker från början har skötts i mindre skala utan ett ordningmaskineri runt om verksamheten - både tidiga kunder och Westbahr AB kom in i ett sådant mönster. Det fungerar bara tills företaget börjar växa och det blir tydligt att det behövs struktur och hjälpmedel.

Fristående enskilda telefonsamtal, e-post och framförallt realtidschat har använts för all intern kommunikation gällandes uppkomna problem och styrning av projekt. Konversationer med kund har det heller inte funnits en uttalad struktur eller samlingspunkt för.

Några olika verktyg för support och projektshantering testades med otillfredställande resultat under något år innan framtagandet av Westimate började diskuteras. Det upplevdes som att inget av de befintliga verktyg som kunde hittas var helt anpassat för situationen Westbahr AB befann sig i och de krav som ställdes på ett sådant system. Både gällandes mångsidig funktionalitet och med tanke på framtiden och tänkta integrationer, bland annat mot ett tidsfaktureringsystem.

3.1.3 Westbahr och arbete i projekt

Under tidsperioden denna analys har utförts har företaget utvecklats och det historiken ovan beskriver har inte varit lika påtagligt under flera senare projekt. Samtidigt som det fortfarande har förekommit ett fåtal projekt där det också har gått överstyr med överskridna tidsramar.

När det gäller projekt kan en tydlig separering göras mellan akademiska/teoretiska och rent industriella projekt. De konsultuppdrag Westbahr AB åtar sig tillhör helt den senare kategorin. Några saker som kännetecknar dessa projekt är att det ofta finns ett beroende av andra parter och andras arbete, både tidigare och parallellt. Ständigt föränderliga krav och system/miljöer som hålls levande över längre tid.

Större delen av arbetet utförs på uppdrag åt kunder men även mycket av det interna arbetet utförs naturligtvis i projektform, rent formellt som om Westbahr vore en kund åt sig själv.

Aktiviteter och aktörer [7] är de entiteter som framträder tydligast när man börjar bena ut vad som är inblandat för att täcka upp de flesta fall av projektliknande arbete, som kan tänkas utföras på Westbahr idag och de närmsta åren. De huvudsakliga grundaktiviteterna i de flesta konsultprojekt och så även på Westbahr är planering, utförande och underhåll. Aktörerna kan vara kontaktpersoner från kunden, konsulter och eventuellt projektledare. De här personerna kan samtliga vara resurser som aktivt deltar i processen.

Kunden kan vara delaktig på så sätt att synpunkter och annan kommunikation används ihop med konsulternas anmärkningar och frågor, som om de ingår i samma arbetsgrupp. Kunden kan även vara en mer passiv kravställare som inte följer arbetet från dag till dag.

Grovt sett kan projekten delas upp i:

- Utveckling - Programmering av system, webbsida, tjänst inklusive integrationer, ramverk, hjälpsystem samt konfigurering och installation.

- Miljö - Inköp, planering, drift av både hårdvara, nätverk, systemprodukter och tjänster. Westbahr fyller i vissa fall behovet av en IT-avdelning och då rör det sig mer om ett heltäckande underhåll än ett väldefinierat projekt.

Det tillkommer sedan i princip alltid support för projekten ur de båda områdena. Det har funnits en viss skillnad i att utvecklingsprojekten oftare har ett mer omfattande sammanhang som blir intressant att följa och analysera tidåtgången på. Medan övriga projekt kan innebära en punktinsats som planeras och genomförs utan lika stora

svängrum och överraskningar i tidsaspekten. Andelen supportstöd upplevs dock som högre i IT-miljöprojekten. Den är utspridd och svårt att förutsäga. Det tillkommer uppgifter som antingen är uppenbara och löses enkelt, "tio minuter här och en halvtimma där", eller uppgifter som är diffusa och svåra att tolka. Sådana som det inte går att få något grepp om innan de plötsligt är lösta. Ofta genom ett enkelt ingrepp.

I utvecklingsprojekten samlas istället buggar som inte är akuta och ändringsförslag ihop och genomförs enligt idealet i fokuserade omgångar, så kallade sprints. Det ger en större möjlighet att resonera kring hur upplevelsen av förväntad behövd tid och jämförelsen med den egentliga åtgången ska kunna bli mer kontrollerad.

I många utvecklingsprojekts inledning på Westbahr brukar det finnas eller vara möjligt att strukturera upp en plan. Vilka större delmoment som ingår i projektet och i ungefär vilken ordning de kan utföras. En projektplan kan i klassisk projekthantering modelleras som ett Gantt [32] eller PERT-schema [33]. Men en sådan plan tenderar att bli inaktuell förr eller senare i dynamiska projekt. På Westbahr upplevs det att stora delar, ofta majoriteten, av ett projekts livscykel befinner sig i lägen där en sådan struktur inte gör så mycket nytta. Om det inte läggs mycket möda på att hålla planen uppdaterad.

3.1.4 Hur ska systemet tas fram

Det fanns en entusiasm på Westbahr över att det borde finnas andra sätt som var mer passande att strukturera arbetet på. För att främja dynamiska IT-projekt och ge ett annat stöd mer än klassisk projektstyrning, ett utmanande förslag för att försöka stödja synen på flexibel tidsåtgång och bygga upp en förståelse för estimeringsproblematiken.

En tidig fråga att besvara var om det var ny- eller vidareutveckling som gällde. Det antogs direkt att det bästa alternativet vore att först testa att utveckla ett verktyg från grunden. Valet att inte använda en möjligtvis likvärdig kommersiell eller öppen lösning föll på en viktig punkt. Att det med ett system som blir så centralt för företagets verksamhet bedömdes mycket viktigt att försöka nå friheten det innebär att ha kontroll över vilka framtida tillägg och integrationer som kan genomföras. Det stod klart att det behövdes en lösning som tål att förändras allt eftersom arbetssätten och företaget utvecklas. Det var även av intresse att kunna låta upplåta verktyget till närstående kunder och partners, för enklare syften såsom supporthantering.

Det andra alternativet hade varit att leta efter en plattform att bygga vidare på eller skriva moduler till för den funktionalitet som saknades. Utan att göra en omfattande undersökning om just detta ansågs det omöjligt att kunna hitta en färdig kommersiell eller öppen applikation att utgå ifrån på grund av att det fanns så många egna idéer och en vision med integrering av Westbahrs intranät i Westimate.

Större kommersiella lösningar som Microsoft Project [9] ansågs krångliga med för mycket "overhead" för att användas fullt ut. De vanligtvis mer lättanvända webbtjänsterna kunde ha varit ett alternativ. Men de som kändes till då, som Projektplatsen.se [21] och TicTac Mobile [24], upplevdes inte som tillräckligt täckande av Westbahrs behov för att undersökas så mycket närmare.

Microsoft Project upplevdes närmast som en standard, med stor förankring hos personer i ledarpositioner inom IT-branchen såväl som i andra brancher - där kunderna för Westbahr vanligtvis finns. Det sågs som ett verktyg som många känner till och har erfarenhet av. Därför ansågs produkten vara en måttstock för klassisk projekthantering att jämföra andra metoder med.

I mer utförliga undersökningar efter befintliga verktyg, med alternativa projektmetoder, upptäcktes ett av särskilt intresse i analysens tidigare skede i december 2007, FogBugz

[25]. Sedan hittas även ett annat mycket intressant långt senare, efter att analysen avrundats och systemdesignen var framtagen. Verktyget som upptäckte för sent för att påverka, Devshop [26], uppmärksammades i detta sammanhanget i samband med att de lanserade version 2.0 av sin tjänst sent på sommaren 2008 [27].

Den stora fördelen med FogBugz var att tjänstens mest utmärkande drag, den evidensbaserade schemaläggningen - Evidence Based Scheduling, EBS [4] - var väl dokumenterad och med tilltalande argument som på ytan adresserade Westbahrs arbetsmetoder bättre än något annat verktyg som påträffades. Innan Devshop dök upp, som har precis samma idé som EBS med några annorlunda koncept. Valet föll tidigt på att inspireras av FogBugz, granska verktyget närmare och utforma Westimate så att det skulle gå att implementera en egen variant av den algoritm som låg till grund för EBS.

3.1.5 Hur ska systemet införas i organisationen

Att införa nya verktyg och rutiner i organisationer sker sällan problemfritt. Även om det är en efterlängtad förbättring. Det måste drivas igenom tydligt och stödjas av organisationen samtidigt som sättet det görs på kan ha stora psykologiska konsekvenser för hur väl det tas emot.

Om det som i det här fallet hela tiden kommer att finnas de gamla sätten att göra saker på, som att skicka separata meddelanden som inte förs in i systemet, så krävs det en insats. Finns det samtidigt inga riktlinjer och uttalade krav som hindrar från att undvika det nya verktyget kommer det förmodligen att kringgås ofta. I dokumentationen till FogBugz beskrivs några av direktiven företaget bakom den produkten själva har använt sig av [34].

Det handlar till stor del om att alla kan agera exempel för andra, det blir ett kollektivt ansvar. Just för ett sådant system som Westimate kan omställningen förstärkas genom att programmerare och tekniker väljer att utföra det som finns korrekt inlagt i systemet i första hand och sedan svarar allt mer avböjande på separata mail och telefonsamtal genom att hänvisa till systemet.

Alla kan börja i liten skala med att tilldela de andra uppgifter i systemet, istället för att ringa eller maila, så ofta det inte glöms. Förhoppningsvis tar det inte lång tid innan det alltid görs - när nyttan med att samla informationen har framgått. Som ledare gäller det framförallt att vara tydlig där, de anställda kommer att märka att systemet används. Kommunikation över andra kanaler bör då minska. Man kommer inte ringa eller gå förbi kontoret lika ofta och inhämta information om vad som ska göras när det finns ett naturligt ställe att samla ett ärendes historik.

Det finns, som nämnt ovan, en till viktig aspekt gällandes införandet av Westimate. Chroust [2] skriver att "psychology asserts that an individual exposed to an unknown/ changed environment exposes a certain amount of fear which in turn seems to be one of the major reasons for objecting to the introduction of new technology". Han slår fast att sådana förändringar måste diskuteras öppet ihop med dem det påverkar under processen så att nödvändiga justeringar kan göras innan förändringen drivs igenom på allvar.

Efter att systemet har etablerats och används till grundläggande uppgiftshantering kan övriga direktiv om mer specifika flöden gällandes estimering och uppföljning av tid införas.

3.2 Arkitektur

Från kravspecifikationen framgick det vilken teknisk plattform systemet skulle byggas på. Det skulle vara en webbapplikation som använder Adobe Coldfusion [10] och Farcry CMS [13] ihop med mer känd grundläggande teknik, som HTML, CSS, Javascript och MySQL. Det skulle även undersökas om systemet skulle kunna göra uppgifter tillgängliga för Microsoft Exchange [15].

3.2.1 Grundläggande teknik för webbapplikationer

HTML/CSS [35]

Den dominerande tekniken på internet för att strukturera upp hur allt innehåll, som text, bilder och andra mediafiler, visas och organiseras i webbläsare.

Javascript [35]

En utbyggnad ovanpå HTML/CSS för att i huvudsak kunna hantera inladdat innehåll interaktivt mellan sidladdningar, eller att genom asynkrona anrop på begäran hämta in ny data utan att en sidomladdning behövs.

SQL/MySQL [36]

SQL är en väldigt känd teknik för att systematiskt lagra och bistå applikationer med data ur relationsdatabaser. Genom att formulera relationer, villkor och mycket annat i en fråga (eng: query) enligt SQL-syntax till en databashanterare sker en förändring i databasen och/eller så returneras ett svar med data ur den.

MySQL är en utav många databashanterare som kan användas enligt ovan. Den har liksom de flesta andra en egen "dialekt" av SQL, vilket innebär att det förekommer en liten uppsättning kommandon som kan vara unika för MySQL eller skilja sig från SQL's standard.

3.2.2 Adobe Coldfusion

Westimate är skrivet i Coldfusion, ett webbaserat skriptspråk som kan jämföras med de mer kända teknikerna PHP, ASP och C# i Microsoft .NET, Ruby och Java Server Pages. De är alla produkter som kan leverera HTML-kod eller andra typer av dokument som svar på en webbklients anrop. Beroende på parametrar och begärt databasinnehåll styrs vilken data som genereras vid varje tillfälle.

Coldfusion har funnits länge och genomgått två större evolutioner. Idag är det en Adobeprodukt, innan dess Macromedia och från början togs det fram av Allaire. Redan 1995 kom den första versionen, innan ASP fanns [11] och ungefär samtidigt som PHP offentliggjordes [12]. 2001 köptes Allaire upp av Macromedia. Coldfusion började då skrivas om från grunden till en Java-applikation som körs på en J2EE server - till exempel JRun, JBoss, Tomcat, Websphere, med flera. Denna släpptes under namnet Coldfusion MX året därpå och innehöll många nyheter. 2005 släpptes Coldfusion MX 7 och samma år var det Macromedias tur att bli uppköpta av Adobe. Åter igen innebar ett ägarbyte nya idéer med prestandaoptimering och utökad funktionalitet som följd. 2007 kom Adobe Coldfusion 8 som är den nuvarande versionen [37].

Språket i Coldfusion finns i två varianter. CFML är en äldre notation som skrivs med fristående taggar som påminner om vanlig HTML.

```
<cfset x = 0>

<cfloop index="i" from="1" to="10">
   <cfset x = x + i>
</cfloop>
```

Möjligheten finns även att skriva kod inom cfscript-taggar med mer kortfattad Java-inspirerad syntax.

```
<cfscript>

x = 0;

for (i=1; i<=10; i++) {
   x += i;
}

</cfscript>
```

Taggen cfscript är ett komplement till CFML. Nästan alla taggar har motsvarande funktioner i cfscript, men det finns ett antal som inte stöds. I de flesta fallen beror det på att en tagg kan kräva en viss uppsättning av andra taggar inuti sig. Eller av deras komplexitet då både HTML/CFML och brödtext kan kombineras inuti en CFML-tagg. Till exempel <cfsavecontent> som sparar den text som renderas utav Coldfusion till en variabel.

```
<cfsavecontent variable="x">
   <!--- valfri text eller textgenererande kod --->
</cfsavecontent>
```

3.2.3 Farcry CMS

De flesta och största webbsystemen som har tagits fram i Coldfusion på Westbahr har byggts med hjälp av ramverket Farcry [13]. Det är ett öppet CMS, Content Management System [14], som ombesörjs av företaget Daemon.

Grundidéen för alla CMS är att erbjuda funktionalitet för hantering och publicering av webbinnehåll, såsom nyheter och artiklar. Ofta med tillhörande bilder och dokument. Målet är att förenkla det manuella kringarbete som krävs vid hantering av innehåll.

Andra funktioner som brukar ingå kan vara navigering, sökning, kategorisering, layout genom sidmallar, flerspråkstöd och publiceringsregler.

Farcry har liksom många andra CMS mycket mer funktioner än så, men Westimate använder bara två delar av Farcry. Dessa är djupt integrerade i Westimate:

- FourQ COAPI. En abstraktion av databaslagret. Datatyper definieras upp i typfiler, både enkla värden och mer komplexa som arrayer stöds. En array

definieras som en typegenskap som kallas Array Table och lagras i en egen tabell. Typerna kan sedan via ett enda klick i Farcrys administrationsgränssnitt skapas i databasen. Ändringar genomförs med samma procedur. Objekt skapas, uppdateras och hämtas sedan från databasen genom funktionerna createData(), setData() respektive getData().

- Användarmodellen. Färdiga funktioner för registrering och autentisering av användare, skapning och tilldelning av användargrupper. Några datatyper för detta från FourQ COAPI används i sin grundform eller genom arv i mer eller mindre alla Farcry-applikationer.

3.2.4 Microsoft Exchange Server

Microsoft Exchange [15] omnämns i kravspecifikationen och är en e-postbaserad kommunikationsserver riktad mot företag. Exchange består av en mailserver, en e-postklient och tillhörande programvara för ytterligare samarbetsstöd.

För att använda Exchange fullt ut ansluter man till sitt konto på en Exchangeserver genom en klient, vanligtvis Microsoft Outlook lokalt på sin dator eller Microsoft Outlook Web Access i webbläsaren. I klienten får användarna tillgång till funktionalitet som att dela sina uppgifter, kontaktlistor och kalendrar med medarbetare.

På Westbahr har produkten i viss utsträckning använts till e-post, bokning av möten, personliga kalendrar. Det fanns en önskan att undersöka förmedling av uppgifter från Westimate till Exchange.

I Coldfusion finns det sedan version 8 inbyggt stöd för integration mot Exchange genom en grupp CFML-taggar för att ansluta till en server och nå objekten där. För att behandla just uppgifter behövs en anslutning och sedan används <cfexchangetask> till att skapa, överföra och ta bort poster.

```
<cfexchangeconnection action="open" connection="C" ...
(serverparametrar)>

<cfexchangetask connection="C" action="create" ... (uppgiftsparametrar)>
<cfexchangetask connection="C" action="modify" ... (uppgiftsparametrar)>
<cfexchangetask connection="C" action="get" ... (uppgiftsparametrar)>
<cfexchangetask connection="C" action="delete" ... (uppgiftsparametrar)>
```

En stor nytta med att integrera Exchange är att det är en standard som ligger nära till hands för den majoritet av människor som är vana vid arbete i Microsofts Windows/ Office-miljö. Det finns även de som använder sin kalender och eventuella uppgiftslista flitigt via mobila tekniker ute på fältet. Som vid resor inför kundbesök när dator och fast anslutning inte är tillgängligt.

3.2.5 Tillägg

Följande utomstående skript har används i implementationen av Westimate:

ajaxCFC [16] (Apache Licence, Rob Gonda)
Ett verktyg för att enkelt kunna hantera datakommunikationen mellan asynkrona javascriptanrop, mer känt som AJAX, och komponenter i ColdFusion.

JSDragDropTree [17] (GNU/GPL, Alf Magne Kalleland)
Ett javascript som används för att visualisera ett projekts uppgifter i en trädstruktur, där noderna kan omplaceras genom enkla musrörelser.

JSCalendar [18] (GNU/GPL, Mihai Bazon)
Ett javascript som visar en konfigurerbar kalender där bland annat datum kan markeras och där egna javascript-händelser kan triggas när ett datum väljs genom att klickas.

Resource Locale Manager (Westbahr, Daniel Niklasson)
Ett verktyg framtaget på företaget för att behandla uppsättningar av gemensamma strängar för olika språk.

3.3 Projekt

3.3.1 Definition

Vad är ett projekt? Enligt Project Management Institute (PMI) är ett projekt en tillfällig insats för att skapa ett unikt resultat [7]. Med tillfällig menas inte att det behöver ske under en koncentrerad tid, utan bara att insatsen har en bestämd början och slut. Slutet kommer när målen har nåtts eller när det blir uppenbart att de inte kommer att nås eller att de inte behövs längre.

Det unika resultatet som ska komma ur ett projekt kan vara en produkt eller service. Det kan även lika gärna vara något teoretiskt som bara finns på papper. Det unika i resultatet syftar mer på projektet i sig, så även om resultatet är likt något som redan finns så kommer det just denna gång på grund av ett projekts unika förlopp att ha nåtts på ett särskiljande sätt.

PMI nämner även i sin definition att det vid sidan av projekt i en organisation finns så kallat "operational work", för att stödja och upprätthålla verksamheten. Upprätthållande arbete definieras som konstant pågående och med repetitiva resultat istället för ett unikt. Arbetet har heller inget mål som avslutar processen när det har uppnåtts.

De grundpelare som arbete i ett projekt utgörs av är precis de samma som för upprätthållande arbete, resurser och aktiviteter. Ett projekt består av en eller flera resurser som utför arbetet och hanteringen av projektet. Insatserna görs i form av aktiviteter där kunskap, färdigheter, verktyg och tekniker används för att möta projektets krav. Projekthanteringen kan vara uppdelad i olika processer som planering, utförande och övervakande.

Upprätthållande arbete som utförs åt en kund kan tyckas hamna i gränslandet mellan definitionerna av projekt och "operational work". Om begreppet projekt används för att hantera upprätthållande processer som inte stämmer helt överens med PMI's definition. Det är inte uppenbart att detta arbetet skiljer från stödjande aktiviteter som utförs i ett projekt. Det är värt att klargöra att i detta sammanhanget tolkas "operational work" att ligga utanför ständigt pågående processer som inte anses vara tillfälliga eller med avslutningsmål för att stödja en kund.

För att förenkla det ovanstående kan en mer vardaglig definition av projekt göras. Som stämmer bättre överens med verkligheten i detta fall. Ett projekt i Westimate behöver inte vara mer än ett sammanhang att placera relaterat arbete i. Det finns alldeles för många olika sätt att organisera upp arbete på för att kunna hitta en smalare definition som är enkel att enas om.

3.3.2 Tidsramar och kostnader

Klassiska ingenjörstekniska projekt såsom exempelvis byggnadskonstruktioner brukar kunna planeras och genomföras inom bestämda tidsramar. I många branscher hör stora överraskande förseningar till undantagen. Tänkbara anledningar kan vara att det används beprövad teknik som har mognat, många moment är upprepningar och projektet kan utföras isolerat från yttre faktorer.

När det gäller projekt inom IT-branschen är det nästan alltid nya förutsättningar som gäller. I mjukvaruprojekt kan sådant som man själv eller andra har gjort innan ofta kopieras och återanvändas i form av designmönster, ramverk, öppen källkod och skript. Det som kommer sedan innebär i princip alltid en unik situation, en helt ny vinkling behövs med okända problem som följd. Det som blir över efter att förpackade lösningar och enklare nyutveckling har löst uppgiften så långt det går är diffusa uppgifter kring integration och optimering, som är mycket svårare att få kontroll över och se alla dimensioner av innan arbetet väl är utfört.

Det finns väl dokumenterat genom otaliga exempel att mjukvaruindustrin lider av ständiga förseningar till större eller minst lika stor del som någon annan bransch. Det går naturligtvis att finna detta i framkanten av många andra branscher, där det under liknande förutsättningar uppstår samma problematik som beskrevs ovan.

Den överlägset största kostnaden i många IT- och konsultprojekt är arbetstiden. Ihop med resurser i form av anställda/konsulter och uppgifter kan det vara den faktor som närmast uteslutande styr ett projekts kostnad. Sådan är situationen för Westbahr AB. Det är sällan materialkostnader eller andra icke tidsrelaterade utgifter spelar större roll i ekonomin än värdet på den tid som går åt. Konsekvensen av svårigheterna med att styra projekt blir således att de kostnaderna springer iväg.

Projektplatsen [21] genomför undersökningar bland dem som använder deras kända projekthanteringstjänst. I den senaste barometern från hösten 2008 [22] valdes 1700 användare ut varav drygt 300 svarade. IT/Data/Telekom är den största branschen för användarna med 22,3% men 37,7% angav att de huvudsakligen arbetar inom IT. De projekt som utfördes var i första hand utvecklingsprojekt (20,1%), systemutvecklingsprojekt (15%), affärsutvecklingsprojekt (12,2%) och kundprojekt (10,4%).

I undersökningen från 2007 anges att 40% av projekten försenades och att precis lika stor andel av projekten överskred sina budgetar [23]. I siffrorna från 2008 är det istället 25% som inte levererats på utsatt tid medan budgeten överskreds i 20% av projekten.

I jämförelsen mellan försenad leverans och överskriden budget förändrades statistiken mycket på ett år. Att gå från 40-40 till 25-20 procent är en märklig förändring som antagligen beror på urvalet. Dock följer siffrorna varann. Detta är inte något starkt bevis men det är svårt att hitta andra exempel att styrka sambandet från. Därmed görs ändå här antagandet att en överskridning av kostnaderna kan likställas med en överskridning av den förväntade tidsåtgången i de flesta projekt som är intressanta här.

Utan fastprisavtal blir det dyrare för kunden. Intäkterna blir i projektet större för konsultfirman, men till priset av att projektet kan anses misslyckat med en missnöjd kund som följd. Internt påverkas även kommande arbete negativt. Det drabbar företaget genom att låsa upp tid som inte kan utnyttjas till att driva igenom andra projekt, eftersom kompetensen är kvar i projekten som drar över. Följden blir ett ökat planeringsbehov och ställtider vid övergångar mellan uppgifter från vitt skilda projekt.

3.3.3 Styrning och behandling av projekt

För att behandla projekt behöver det utföras ett visst arbete för att hålla ihop projektet och dess uppgifter, resurser och arbetstid. Vad man sedan kallar detta är inte så intressant att reda ut i detalj här. Projektstyrning, projekthantering och projektledning är alla olika termer som används. Som många av egna erfarenheter tolkar olika och blandar ihop. Styrning och ledning innebär till stor del samma sak och brukar oftast fokusera på att utföra arbetet rätt. Projekthantering kan då sägas handla mer om att utföra det rätta arbetet.

I framförallt större projekt är en projektledare ett måste för att hålla ihop och fördela arbetet. På Westbahr AB hör det till ovanligheten att fler än fem personer är inblandade i ett projekt, oftast rör det sig om upp till tre. En explicit projektledarroll behövs då inte alltid utan en av de anställda resurserna agerar vanligtvis som ledare också. Det kan vara som så att någon resurs är den enda som är involverad och utför arbetet, medan den som ansvarar för projektet inte utför några arbetsuppgifter utan bara följer processen. I projekt för allmänt internt arbete anses alla vara involverade men där är det inte heller säkert att det behövs en traditionell projektledare.

3.3.4 Styrningsmodeller

För styrning eller ledning av projekt (eng: project management) finns det många modeller som brukar följas för att konkret strukturera förloppet och arbetsmetodiken. Några kända heltäckande modeller är PRINCE2 [28], RUP [29] och Scrum [19]. Dessutom finns det mindre omfattande modeller, som är inriktade på tidsbedömningar och leveranser utifrån hur projektet delas upp i faser och delmoment, som Critical Path Method, CPM [30], Work Breakdown Structure, WBS [31] och Program Evaluation & Review Technique, PERT [33]. Innehållet och syftet med några av modellerna kommer att beröras senare med inriktning mot hur projekt ska kunna estimeras.

3.3.5 Tidsstyrning

Inom projekthanteringsområdet finns det gott om termer och teorier för hur tidsfaktorer ska behandlas. Det är naturligt eftersom det finns oändligt många sidor av hur projekt organiseras och styrs. Forskningen har belyst många av dem ur olika synsätt. Deadlines, milstolpar, att kommunicera status löpande och flera andra sätt att främja planering och upptäcka behov av åtgärder. Att kunna göra anpassningar i planeringen när kontroller som sker genom styrningen uppfångar motgångar.

Det blir sedan lätt en fråga om mänskliga faktorer. Hur olika sätt att tidsstyra projekt påverkar beteende och därmed resultat i olika projektfaser och den övergripande effektiviteten.

Steve McConnell väver ihop detta med hanteringen av estimat genom att hävda att en bra användning av estimat ger en tillräckligt klar bild av projektets verklighet för att möjliggöra för projektets ledare att ta beslut om hur projektet ska styras för att nå tidsmässiga mål [1].

3.3.6 Struktur och stöd

Det finns ett par givna abstrakta områden som alltid är av stor betydelse när det kommer till att hantera arbete i projekt. Vilka strukturer som uppgifter och annan information organiseras efter och vilka stöd som behövs för att underhålla strukturen

genom arbetsflödet. För projekt av betydande storlek har detta under många år nu hanterats i datasystem. Med teknikens hjälp tillkommer stora möjligheter att utveckla hjälpmedel.

För att nå rätt information kan en uppdelning göras mellan fysisk och logisk strukturering av uppgifter. Med den fysiska menas en arkitekturell uppbyggnad i hierarkier som vanligtvis visualiseras som träd där ägande eller liknande unika band mellan objekt avgör ordningen. Det andra så kallade logiska sättet att ordna uppgifter är istället associativt där ett tillfälligt urval görs efter söktermer, kategorier, status, tilldelningar eller vad som nu finns tillgängligt.

Information i projekt består inte bara av en uppdelning av arbete i uppgifter. De sammanhållande medel som sedan vanligtvis brukar användas är bland annat filer, dokument, referenser till fysiska objekt i verkligheten och olika sorters kommunikation, som nyheter, e-postmeddelanden och kommentarer.

Det behövs utöver dessa medel även kognitiva hjälpmedel som ger stöd åt hjärnans funktioner och binder ihop allt till ett flöde. En projektledare och/eller ett system hanterar detta genom scheman, påminnelser, prioriteringar, stöd till initiativ, historik, eskaleringar och inslag av tidsstyrning. Information kan även behöva framhävas eller döljas, till exempel när den är ny eller föråldrad.

3.4 Estimat

3.4.1 Att estimera

Ett estimat är en värderad uppskattning om något mätbart [1]. Uppskattningen ska så gott som möjligt baseras på erfarenheter, kunskap och tillgänglig information. Detta för att det helst ska bli en uppskattning som har mer substans bakom sig än en ren gissning. Men det är givetvis inte säkert att ett estimat som framstår som bra kommer närmare resultatet än en grundlös chansning [2].

Utfallet av det som estimeras ska vara oklart i stunden då estimeringen görs. Det är uppenbart att det är så i de fall där möjligheten att förutbestämma resultatet saknas. Ibland är dock inte detta sant. I vissa fall går det verkligen bara att chansa och det är inte mycket att göra åt. I sällsynta fall kan den som estimerar med mycket hög sannolikhet avgöra precis vad resultatet kommer att bli. Det är inte heller mycket att göra åt. Dessa båda fall definieras som estimeringar för att förenkla konceptet.

I det första fallet ovan kan estimeringen nog ändå anses vara något mer än en gissning eftersom alla estimerar på sitt egna vis. En del tar ofta i rejält medan andra tenderar att vara lagda åt det optimistiska hållet och i regel underskatta. Så även i gissningar kan det finnas ett visst personligt mått. Personen som sätter ett estimat kan nog själv uppleva att det görs med mer slump än fakta som grund, men det är nog ofta som mer än slumpen spelar in. En enskild estimering är på sätt och vis alltid en gissning och den kan vara obetydlig på egen hand. Men det bör antas att den får betydelse i förlängningen totalt sett. I sammanhang där större mängder estimeringsdata samlas in, bör det gå att få indikationer som är bättre än gissningar, genom att finna mönster eller förändringar som framträder ur en serie av många estimat [4].

I de fall en enskild estimering är väldigt viktig är det inte längre så enkelt att börja argumentera för att mycket av det som ligger bakom jämnar ut sig och kanske inte spelar så stor roll i helheten. Estimeringar kan inte bara värderas utan även tolkas på en mängd olika sätt. En uppskattning kan vara medvetet optimistisk eller pessimistisk. Den

kan kommunicera en uppfattning eller symbolisera en ståndpunkt som inte delas av alla inblandade. Det går inte att veta hur estimeraren har tänkt när man får se ett estimat från någon annan. Ibland kanske det behövs en motivering ihop med uppskattningen.

Ett estimat kommer att motsvaras av ett resultat när det som skulle mätas har avgjorts. På så vis kan uppskattningar följas upp genom en jämförelse med resultatet. I det enskilda fallet kan det vara nödvändigt att noggrant notera den verkliga arbetsinsatsen ihop med påverkan av oförutsedda händelser. Då framgår det hur väl uppskattningen överensstämde med utfallet. För att avgöra hur bra ett estimat var finns det inget annat att gå efter än hur väl utfallet stämde överens med uppskattningen. Bra estimat är helt enkelt det samma som överensstämmande estimat [2].

Estimat kan inte göras bättre än kraven de ställs mot och förutsättningarna de gjorts under. Det som ska estimeras behöver först förstås och alla inblandade kan också behöva dela denna förståelse [2].

3.4.2 Relevanta entiteter för estimat

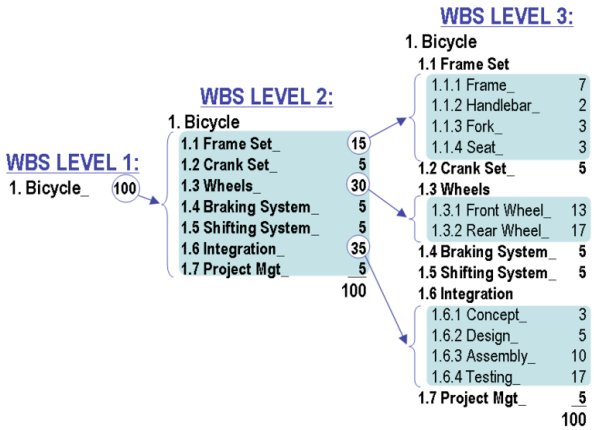
De entiteter som detta arbete undersöker är uppgifter, projekt och de personer som är resurser i projekten. Det är intressant att reda ut hur goda möjligheterna egentligen är att estimera uppgifter och projekt, vilket stöd som kan ges, samt vad det kan tänkas gå att få fram för resultat om resurser som kontinuerligt estimerar sitt arbete.

3.4.3 Att estimera ett projekt

Att i ett enda steg estimera ett helt projekt är givetvis inte rimligt [1]. Att uppskatta en uppgift i taget är enklare [4]. Det man kan resonera vidare kring är om estimat av ett projekts uppgifter kan korreleras mot en total uppskattning av projektet.

Ett projekt består av dess uppgifter och organisatoriskt kringarbete. Allt som hanteras utanför uppgifterna får lämnas utanför här. Men den spontana uppfattningen är att en erfaren projektledare bör grovt kunna avgöra mängden kringarbete efter projektets storlek och karaktär, eventuellt är hanteringen proportionell mot antalet uppgifter om det är en särskild mängd administration som krävs per uppgift. Från projektets komplexitet blir det då ett samband där en faktor kan läggas till för att få den totala tiden inklusive hanteringen av projektet [3].

Ett synsätt som ofta har används för att definiera ett projekts helhet genom en uppdelning i mindre delar är WBS, Work Breakdown Structure [31].



Figur 3.1: Work Breakdown Structure.

Strukturen börjar med projektets helhet eller absoluta slutmål som högsta nivå. Arbetet bryts sedan ner i grupperade undernivåer av delmål i form av arbetspaket där summan av en nivås alla delar måste vara lika med nivån över. Denna princip kallas 100%-regeln och meningen är att projektets omfång ska tydliggöras, inte att bena ut ett fullständig lista utav uppgifter eller en klar projektplan.

Om man bara tittar på tiden som omfattas av uppgifterna borde antagandet att ett projekt är lika med summan av sina delar stämma bra. Om ett projekt då finns definierat och uppdelat i en mängd uppgifter som är estimerade borde deras totala värde motsvara en estimering av hela projektet.

Dock är det så att de projekt som Westimate ska användas till väldigt sällan fixeras i en projektplan, utan stora tillägg och revideringar. Med undantag för väldigt små projekt som är avstyrda på någon dag. Hela projektets livscykel definieras inte på en gång tidigt i processen. Från tidigare erfarenheter är det snarare så att det är de sista kraven som tillkommer som oftast tar större delen av konsultprojektens tid i anspråk. De lyckas väldigt ofta döljas av projektets fria komplexitet.

Projekten är levande och förändras vilket leder till att det nog behövs ett annat synsätt. Istället för att fästa sig vid hela processen och den totala tiden borde det kunna vara mer genomförbart och hjälpsamt att räkna på tiden det tar att lösa de för stunden kända uppgifterna. Som komplement till detta bör det gå att granska i vilken takt uppgifter tillkommer, eller betas av. Med förhoppningen om att osäkerheten över vad som ingår i projektet ska minska och att trender i ökande eller avtagande arbete säger mer om i vilken riktning projektet rör sig. Det upplägget skulle kunna vara bra anpassat till verkligheten Westbahr befinner sig i.

En hypotes kan nu formuleras. *Enskilda uppskattningar av ett projekts kända uppgifter bör korrelera mot en total uppskattning av hela projektets kända återstående tid*. Ett rörligt mått som förändras allt eftersom och inte ger något absolut besked men som samtidigt bara belyser det som är känt. Mer om detta följer under det kommande avsnittet om metoder för projektestimering.

3.4.4 Att estimera en enskild uppgift

En estimering av en uppgift kan göras på ett par olika sätt. Det naturligaste måste vara att uppskatta antalet arbetstimmar som uppgiften kommer att kräva. Då och då kan det vara något annat attribut som är mer intressant. Till exempel vilket datum som arbetsuppgiften beräknas vara klar. Dessa attribut har tidsaspekten gemensam. När uppskattningen görs har estimerarens tidsuppfattning en viktig roll.

Det finns också icke tidsrelaterade attribut. Som måttet på hur stor andel av en påbörjad uppgift som har utförts, rent resultatmässigt, eller en uppgifts komplexitet.
Komplexiteten borde inte vara direkt relaterad till tiden uppgiften tar att utföra, eftersom en expert bör kunna lösa uppgifter som anses ha hög svårighetsgrad på kort tid. Det blir mycket väl en lika tidskrävande insats som att lösa en enklare men mer repetitiv uppgift [1].

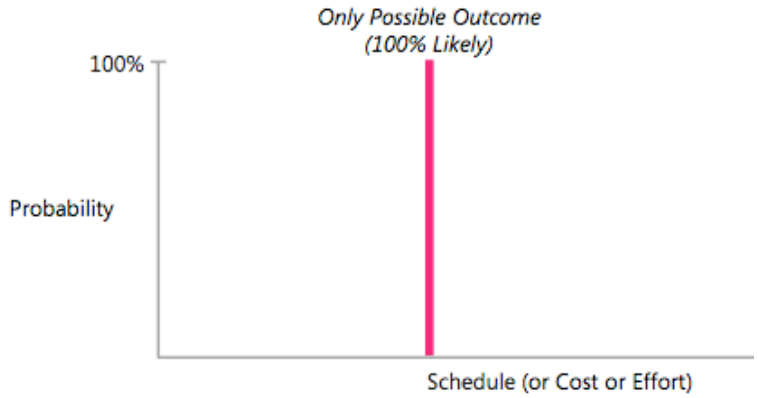
Det kan upplevas som naturligt att i första hand estimera uppgifter innan de påbörjas, eller i alla fall tidigt i processen. Men uppgifter har en livscykel där estimat egentligen kan läggas när som helst. Detta kan vara av betydelse vid tolkningar eller användning av estimeringsdata.

Estimat kan även förändras under tiden en uppgift utförs. Ny information, oväntade framsteg eller motgångar kan göra att den aktuella estimeringen inte stämmer lika bra

längre. Om uppskattningen istället hade gjorts i stunden efter den uppenbara förändringen hade estimatet varit annorlunda. Eftersom ett estimat alltid kommunicerar sitt värde och därmed har en påverkan i arbetet är det nog så att det gärna ska hållas uppdaterat vid en förändrad situation.

Möjligheten finns att spara all ändringshistorik. Det kommer säkert i enstaka fall vara intressant att gå tillbaks till en uppgift och se hur uppskattningen förändrades. Det ursprungliga estimatet kan vara intressant framförallt för egna jämförelser. Men på projektnivå, som vid sammanräkningar, blir det för detaljerat att ta hänsyn till all historik. Det ger inget genomslag som är praktiskt användbart [3].

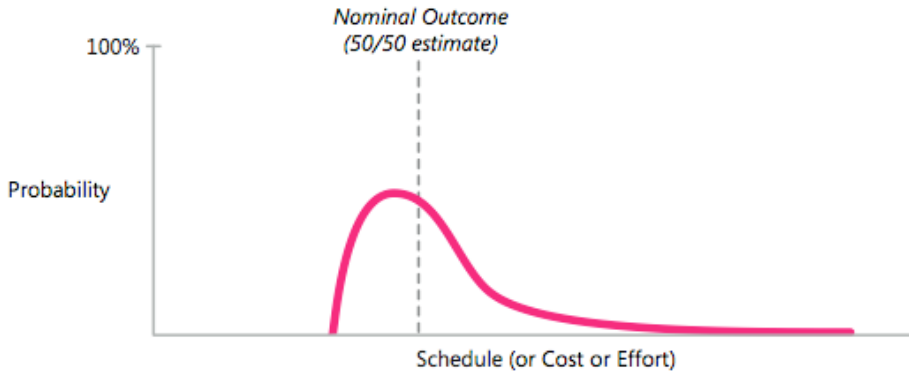
Att använda det senaste estimatet duger bra eftersom de tidsspann där gamla estimat gällde inte längre är aktuella. Det skulle behövas parallella uträkningar som beskriver utfall som på grund av ändringen blir paradoxala. Ett estimat på arbetstimmar ska inte kunna ändras till kortare tid än vad som redan har utförts. Övriga uppdateringar är logiska och leder alltså till att det bara är den senaste estimeringen som är intressant utåt, utanför en uppgifts egna interna information.



Figur 3.2: Endimensionellt estimat.

Med undantag för tankeexemplet med inaktuella estimat i stycket ovan har vi hittills bara granskat endimensionella estimat. Där ett enda värde beskriver uppskattningen, se figur 3.2. Men det kanske är för oflexibelt i många situationer, och signalerar mer ett mål än ett estimat. Vad ett estimat symboliserar kan vara viktigare än just de värden som valts.

Istället för att ange ett estimat med ett enkelt värde åt gången, går det att välja mer avancerade modeller. Varför inte en sannolikhetsfördelning? Med en graf som beskriver vilka uppskattningar som upplevs motsvara sannolikheterna för deras utfall mellan 0 och 1 - eller 0 och 100 procent - på axeln [1], se figur 3.3.



Figur 3.3: Sannolikhetsfördelat estimat.

En förenklad variant av detta är att uppskattningsvärden även kan anges i ett intervall, till exempel tio timmar plus minus tre timmar. Eller att en mindre serie av några värden tilldelas olika vikt, t ex två- och trepunktsestimering [1]. Ofta uppstår det situationer där det finns god anledning att ange två extremvärden för bästa och sämsta tänkbara utfall för att illustrera dynamiken i uppskattningen. Ett tredje värde, som ofta används vid estimeringar av denna typen är det mest sannolika utfallet, det vill säga det som skulle angetts om bara ett värde skulle använts.

Det finns gott om förslag på hur dessa trepunktsestimat kan kombineras till ett enda värde genom enkla viktade omräkningar, detta kan exempelvis göras i Microsoft Project [9]. Ett sådant värde är användbart om det ska gå att summera serier av estimat enkelt. Detta är såklart alltid spekulativt. Vid varje sådan sammanräkning får det helt enkelt bedömas om metoden kan upplevas rättvisande.

Alla uppgifter måste brytas ner i hanterliga bitar, det ska handla om arbete i storleksordningen timmar och inte dagar. För kan man inte ange deluppgifter timvis har man inte tänkt igenom uppgiften och vad den innebär. Då är det inte lika stor mening med att estimera eftersom att osäkerheten blir för stor [1] [4].

3.5 Metoder för projektestimering

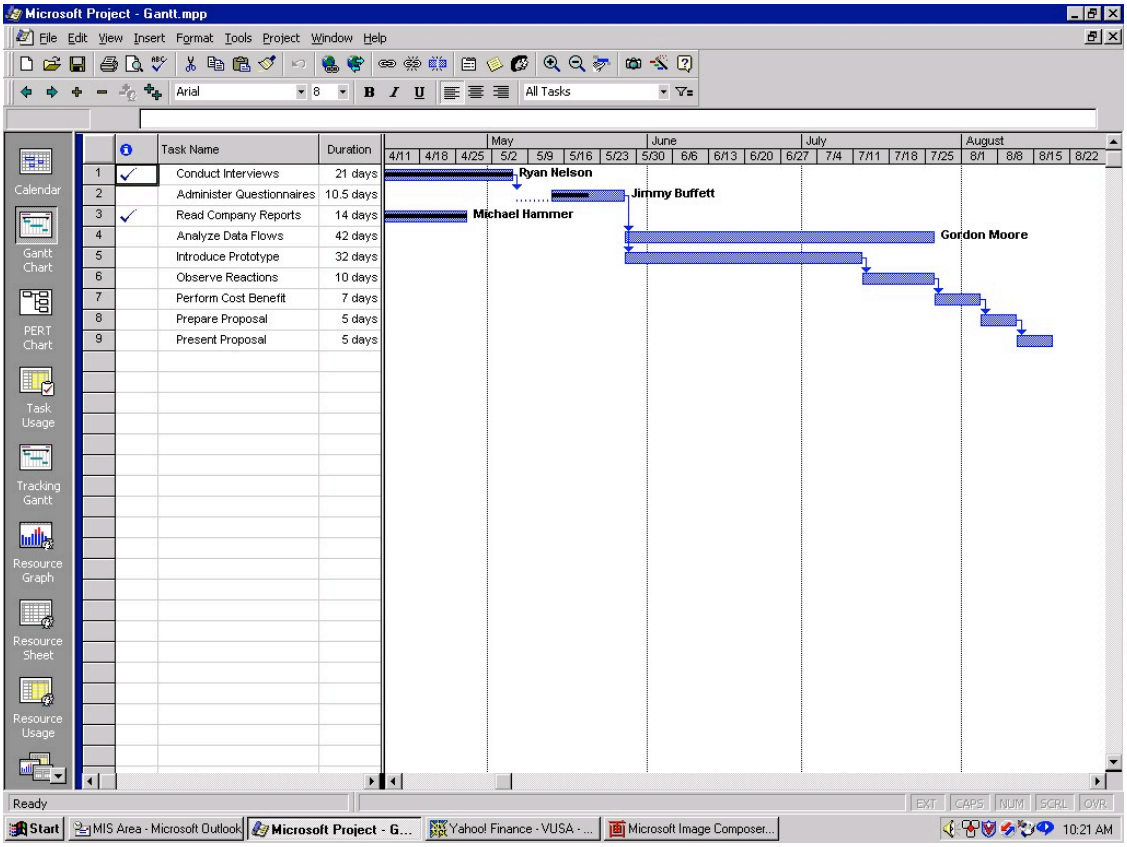
Det har behandlats hur en enskild uppgift kan estimeras och även några grundläggande hypoteser och resonemang kring att ett projekt kan ses som summan av dess uppgifter. Syftet här är att redogöra för några av de metoder som har identifierats för att estimera projekt i helhet. Med detta menas att sammanställa och visualisera ett mått på exempelvis vad som är avklarat och vad som återstår, i bästa fall projektets återstående tid. Moderna metoder brukar även granska projektets riktning, genom att följa hur ett sådant mått förändras, för att visa om det finns indikationer på att projektet går mot eller ifrån ett avslut.

Detta avsnitt inleds med en kort genomgång av två klassiska metoder, Gantt och PERT, som har använts i många större kommersiella projekthanteringssystem, däribland Microsoft Project. De är troligen de två mest kända sätten att visualisera ett projekts förlopp. Slutligen beskrivs en relevant del av Scrum samt EBS, Evidence Based Scheduling.

3.5.1 Gantt

Ett Gantt-schema [32] över ett projekts uppgifter, eller aktiviteter som de ofta omnämns i de fall där det är en fas eller en större uppgiftsamling, visar på vilka aktiviteter som projektets resurser kommer att utföra för att nå fram till projektets slutförande. Projektets status framgår såtillvida att det syns hur mycket av aktiviteterna som har utförts och hur mycket som återstår.

En projektplan görs oftast upp i grova drag först och fylls på med detaljer på lägre nivå. Vid större ändringar i komplexa projekt kan mycket omsorg behöva läggas på att tolka och strukturera om schemat.



Figur 3.4: Exempel på ett Gantt-schema i Microsoft Project.

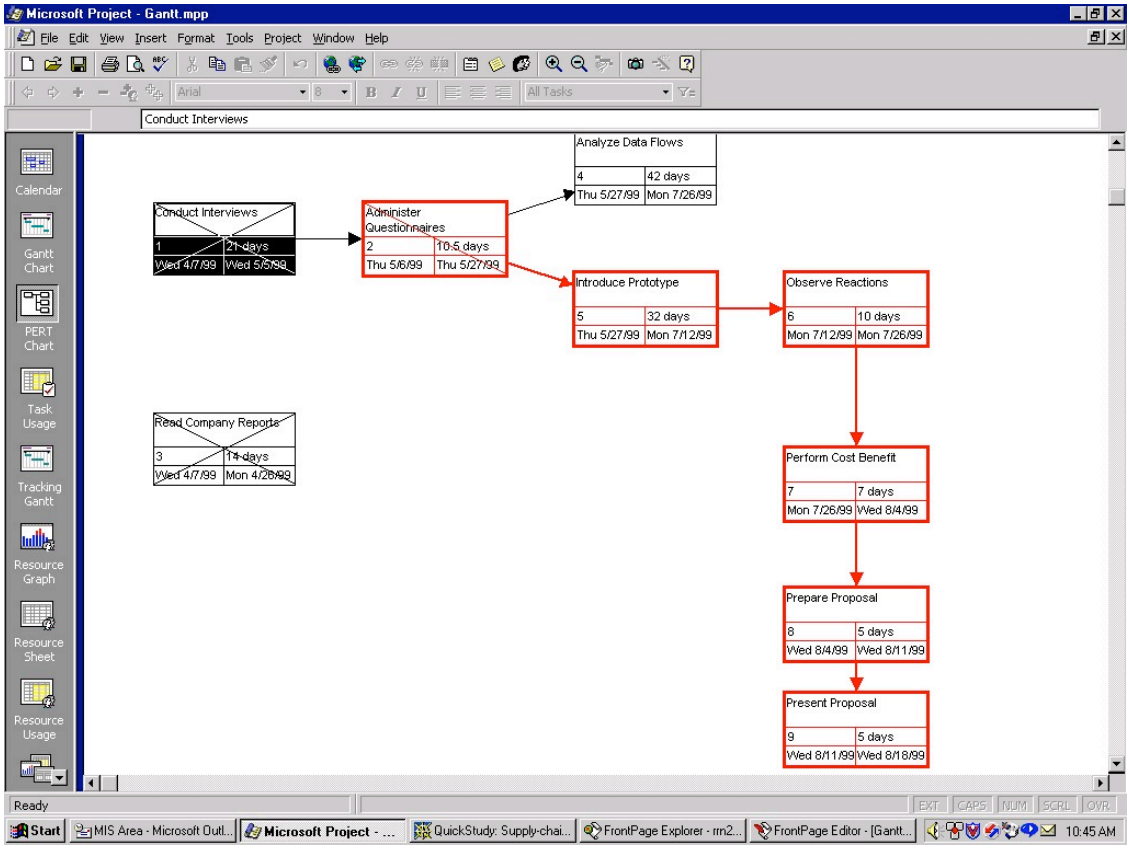
Ett problem är att aktiviteterna definieras som beroende av varandra i förenklade steg som sällan motsvarar hela verkligheten. Detta leder till att det i schemat inte alltid går att se exakt hur resterande förlopp påverkas av att en aktivitet försenas. Inom modern systemutveckling existerar det i många fall väldigt få starka beroenden mellan mindre grupper av uppgifter [5].

Det främsta användningsområdet för Gantt-scheman i många situationer är att använda det på en högst överskådlig nivå för att kommunicera en planering i grova drag inför insatser, men inte för att hålla reda på projektets alla detaljer och bygga ut schemat med sådant som tillkommer efter hand.

3.5.2 PERT

Program Evaluation Review Technique [33] är en modell för att visualisera hur uppgifter i ett projekt hänger samman, dels för att med dess hjälp kunna göra tidsbedömningar av projektets slutförande.

PERT används ofta ihop med Gantt-scheman då de kompletterar varandra bra, eftersom PERT-grafer lätt blir ännu mer komplexa och därmed svårtolkade för större projekt.



Figur 3.5: Exempel på en PERT-graf i Microsoft Project.

Noderna i PERT-grafen kan använda trepunktsestimering för att göras mer dynamisk. Det anges då för delmomentets tid det sämsta fallet, det bästa fallet och det mest sannolika fallet. Grafen kan sedan visas på olika sätt utifrån dessa fall, samt att en viktning av dem kan göras för att visa ett enda gemensamt förväntat förlopp.

Inom traditionell projekthantering och till exempel med Microsoft Projects grundinställning för trepunktsestimering tas värdet fram enligt viktningen:

E = (Bäst + 4*Sannolikast + Sämst) / 6

Standardavvikelsen blir V = (Bäst-Sämst) / 6, med andra ord är den starkt viktad åt det sannolikaste estimatet. Det hör till att ställa in koefficienterna efter vad som motsvarar tidigare erfarenheter mot det aktuella projektets karaktär. Enligt Steve McConnel är det intuitivt mycket vanligare att många, när de anger ett enda estimat, hamnar mycket närmare bästa fallet än det omvända [1]. För att få ett mer pessimistiskt mått kan exempelvis följande viktning användas istället:

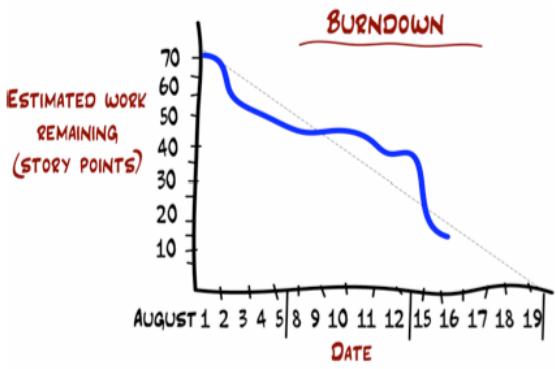
E = (Bäst + 2*Sannolikast + 3*Sämst) / 6

Genom att begära det i Microsoft Project görs en analys av grafen enligt medelvärden som fås ur den valda viktningen. De olika måtten kan även slå igenom på grafen i en separat analys för att visa när projektet kommer att slutföras för de tre olika fallen - det kortaste, det mest sannolika och det längsta förloppet.

3.5.3 Processdiagram enligt Scrum

Scrum [19], som är av de mest kända agila utvecklingsmetoderna, använder sig utav processdiagram (eng: burndown chart) för att visualisera hur ett projekt eller en etapp (eng: sprint) av utvalda uppgifter betas av.

Tanken är att det finns en mängd uppgifter som är utvalda att utföras och att de bedöms, antingen genom ett poängsystem för hur komplexa de är eller genom att de tidsuppskattas.



Uppgifterna samlas i en så kallad backlog där de markeras som avklarade efter hand och processdiagrammet ändras dagligen efter hur många uppgifter som har klarats av, tillkommit eller plockats bort från backlogen.

Metoden kan tillämpas som ett visuellt hjälpmedel i att estimera om projektet går mot ett avslut och när projektet eller etappen kan vara klar. Oftast finns det ett bestämt mål som backlogen anpassas efter [20].

Figur 3.6: Processdiagram.

Scrum och andra relativt moderna metoder brukar använda faktorer för att justera estimaten på uppgifter så att det finns ett justerat värde som används i beräkningar vid sidan av det ursprungliga. I Scrum används vanligtvis en konstant komplexitetsfaktor som bestäms per projekt baserat på hur mycket det aktuella projektets karaktär bedöms kunna minska arbetslagets produktivitet [19].

3.5.4 Evidence Based Scheduling

De flesta traditionella projekthjälpmedel är väldigt enkla och lämnar all tolkning åt projektledaren. Scheman genom Gantt och PERT-metoderna använder de estimat som anges rakt av utan någon inbyggd kritik eller återkoppling såsom historiska data.

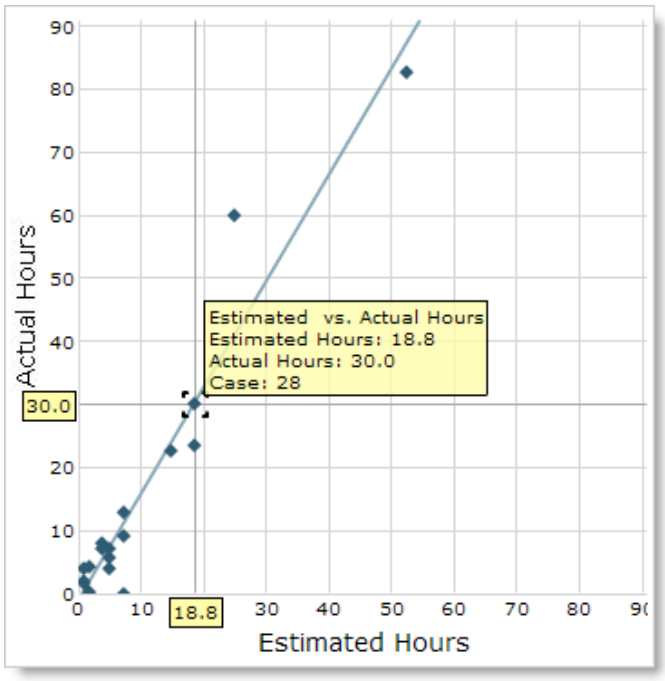
Evidence Based Scheduling, EBS [4], är en metod framtagen av Joel Spolsky som presenterades under hösten 2007. På företaget FogCreek i New York har bugtrackerverktyget FogBugz [25] utvecklats med denna metod och en algoritm som central del.

Tekniken går ut på att försöka förutsäga en sannolikhetsfördelning för när ett projekt kommer att vara klart. Estimat och scheman i beräkningarna justeras efter tidigare resultat. Om alla uppgifter historiskt sett tar dubbelt så lång tid mot vad som först anges, så kommer detta spela in genom att systemet räknar med ett dubbelt påslag från de nya uppskattningar som läggs, och så vidare.

Estimeringar i EBS

Varje uppgift tilldelas en resurs som ska utföra arbetet. Resursen uppskattar hur mycket tid uppgiften kommer att kräva och håller sedan själv eller med en timer i systemet som hjälpmedel reda på den egentliga tiden och rapporterar detta. När uppgiften har markerats som klar läggs det förväntade och det verkliga tidsmåttet till resursens tidbank, se figur 3.7.

Varje resurs får en historik utav värden som visar på hur väl resursen estimerar sitt eget arbete. Förändringar i förmågan att göra detta hanteras genom att det bara är de 200 senaste värdena som sparas. Värdet kallas *velocity* och är den faktor som den verkliga tidsåtgången för en uppgift ska multipliceras med för att få estimatet. För en uppgift som tog dubbelt så lång tid som estimerat är alltså *velocity*-värdet ½ i beräkningarna.



Figur 3.7: Uppföljning av en resurs färdiga uppgifter.

Innan en betydande mängd faktorer har samlats in för nytillkomna resurser fylls tidbanken på med en slumpserie av mestadels dåliga faktorer. Ett positivt genomslag ska vara särskilt osannolikt när historik saknas.

Timmarna som det räknas på kan antingen vara ren och skär arbetstid hårt kopplad till uppgifter, eller precis all tid. Inklusive kaffepauser, möten, telefonsamtal om andra uppgifter och övriga distraktioner. Bara det görs konsekvent. Modellen anpassar sig då efter när uppgifterna kan vara klara utan en exakt fördelning mellan arbetstid och distraktionstid. Spolsky resonerar kring att distraktionerna i slutändan kommer att jämna ut sig och ge samma relativa resultat oavsett vilket av de två synsätten som används.

Monte Carlo-simuleringar

Den metod som används för att genomföra beräkningar i EBS är uppkallad efter det kända kasinot i staden Monte Carlo, i Monaco.

Grundprincipen för en Monte Carlo-metod är att ett problem som är svår- eller olösligt med deterministiska metoder simuleras genom upprepade försök där slumpmässiga val bestämmer de enskilda utfallen [38]. För numeriska problem kan det, baserat på de genomförda försöken, enkelt tas fram en statistisk fördelning och beräkningar av medelvärde och standardavvikelse kan göras. För problem som är korrekt formulerade och där en stor mängd utfall kan beräknas genom datorer brukar resultat erhållas, oftast inom riskhantering för kostnader och scheman, som är mer användbara än mänsklig intuition [39].

Ett konkret exempel på en Monte Carlo-simulering ges här för att förtydliga. Att uppskatta värdet av pi kan göras genom att slumpmässigt generera många punkter

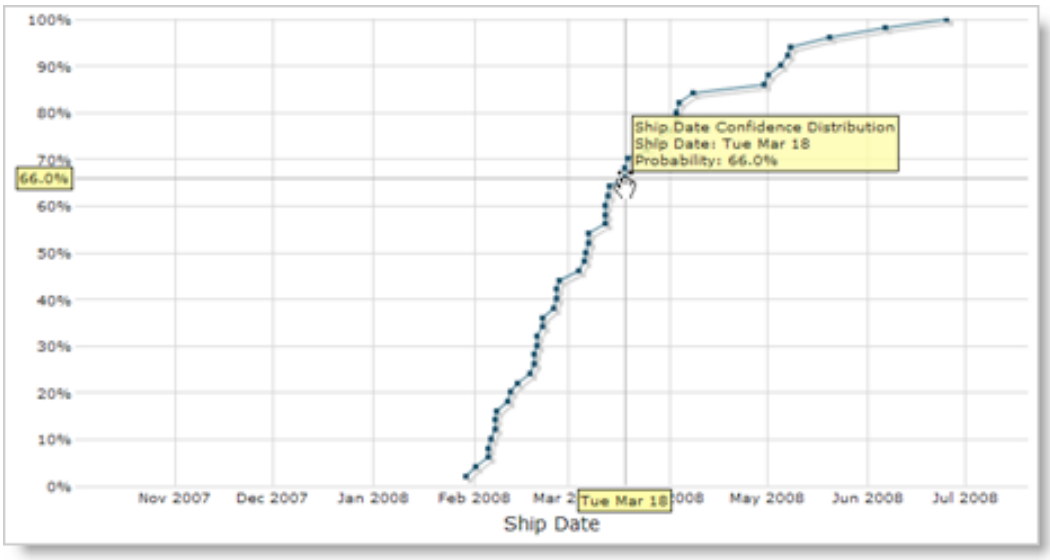
(x,y-koordinater) inom en tänkt kvadratisk yta. En cirkel med radie lika med kvadratens halva bredd kan inrymmas och om avståndet från en punkt till kvadratens mitt överstiger radien hamnar punkten utanför cirkeln. Kvoten som fås mellan punkter som hamnar inom och utanför cirkeln multipliceras sedan med 4 enligt enkel geometri och ger ett uppskattat värde av pi [40].

Sannolikhetsfördelning enligt Monte Carlo

Det huvudsakliga stödet som EBS kan ge är att med historikens hjälp ta fram när projektet ska kunna vara klart. Eller rättare sagt när de ofärdiga uppgifter som har estimerats ska kunna vara klara.

Detta görs genom en algoritm som använder sig av en Monte Carlo-simulering för att slumpmässigt simulera förloppet n gånger och tilldela utfallet av varje omgång sannolikhetsmåttet 1/n %.

I varje Monte Carlo-runda räknas alla återstående uppgifters estimat om med hjälp av en slumpmässig tidigare faktor, för den resurs som ska utföra den aktuella uppgiften. Dessa mer realistiska värden, enligt historiken, justeras mot redan rapporterad tid och summeras sedan till arbetstimmar. Slutligen omformas arbetsinsatsen till ett datum baserat på de inblandade resursernas scheman.



Figur 3.8: Sannolikhetsfördelning för ett projekts slutdatum.

När detta görs 60 gånger, som i FogBugz, framträder en sannolikhetsfördelad graf, se figur 3.8, med ytterligheter i snabbast tänkbara förlopp till vänster med låg sannolikhet och det sämsta simulerade fallet längst till höger med hög sannolikhet för projektets slutförande.

För de fall där så mycket som möjligt simuleras så dåligt som historiken tillåter bör sannolikheten närma sig ett mycket högt värde. Men modellen täcker inte in de oväntade scenarion där det går ännu sämre. Likaså borde det i den andra ändan finnas en mindre chans att arbetet går fortare än det bästa simuleringsfallet. Grafen har med andra ord öppna gränser till 0 och 100% som aldrig nås. I EBS används istället 5 och 95% symboliskt för att lämna ett utrymme för förlopp som simuleringen inte täcker in.

Grafen blir brantare ju bättre och konsekventa faktorer som finns i historiken. Har projektets resurser svårt för att estimera och de inte förbättras över tid kommer intervallet förbli brett och osäkerheten stor - precis som verkligheten då visar.

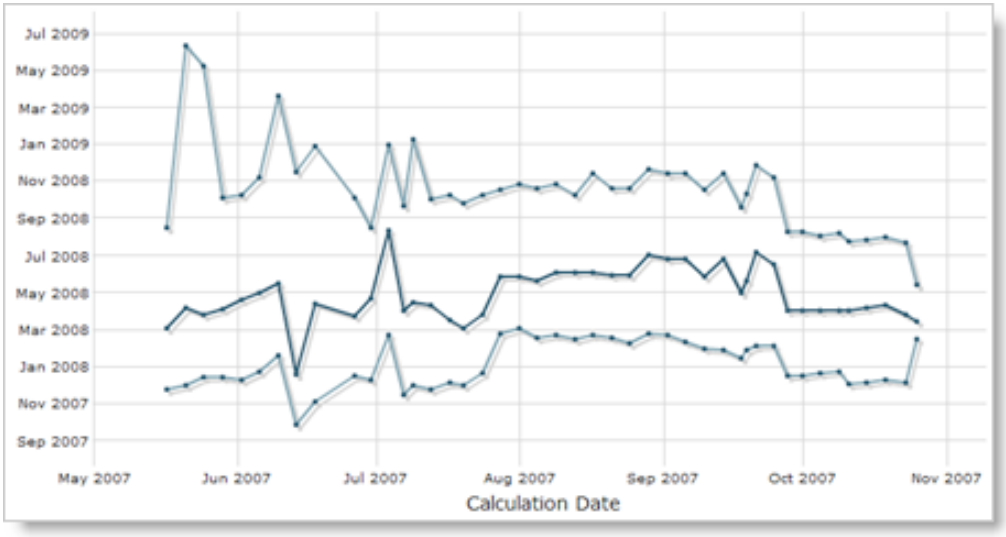
Sannolikhetsfördelningens förändring över tid

Det är inte alltid som ett projekts alla uppgifter finns identifierade tidigt i processen. Ofta tillkommer de efter hand. Det är oftast först när uppgifter betas av fortare än det tillkommer nya som det finns indikationer på att projektet börjar gå mot sitt slut.

Viktig information kan fås genom att varje dag spara aktuella slutdatum för tre sannolikhetsvärden ur fördelningsgrafen, och jämföra hur dessa förhåller sig till varandra över tid. De som FogBugz lagrar är de optimistiska 5%, ungefärligt medelvärde 50% och pessimistiska 95%.

För en stigande graf så skjuts slutdatumet alltså framåt, på grund av att uppgifter tillkommer eller att estimat uppdateras. De ursprungliga estimaten har ju redan räknats om med hjälp av historiken för de tidigare värdena på grafen, så det är bara de fallen som kan skjuta tidsramen framåt. Om grafen istället sjunker betyder det helt enkelt att uppgifter slopas eller att de har överskattats och estimeras om med lägre värden. En rak graf säger då att projektet fortlöper under kontroll.

När de olika graferna konvergerar betyder det att osäkerheten minskar och att slutdatumet närmar sig. I figur 3.9 har vi ett exempel, där den övre kurvan är för 95%, mitten 50% och den nedre 5%.

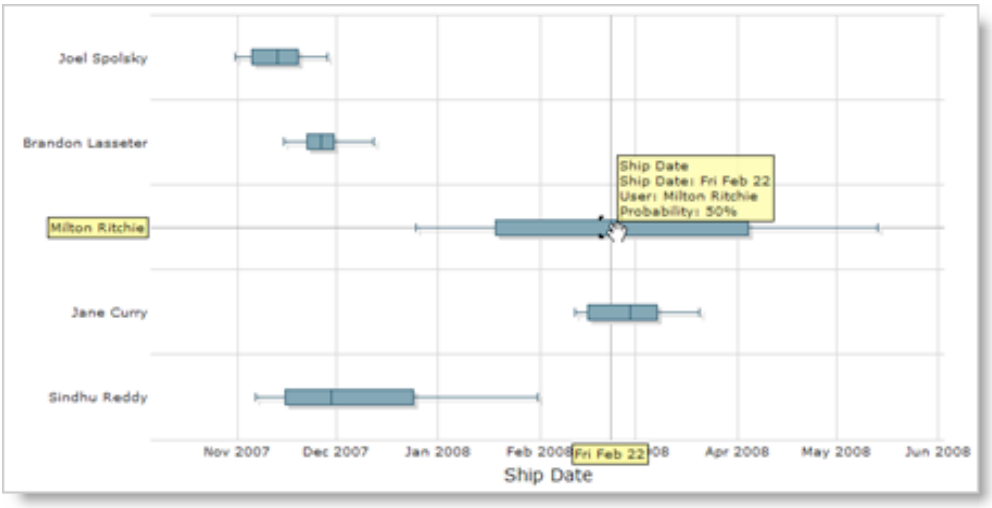


Figur 3.9: Slutdatumets förändring över tid med tre sannolikhetsmått.

Sannolikhetsfördelning per resurs i ett projekt

För de involverade resurserna i ett projekt kan även en sannolikhetsfördelning per resurs tas fram, se figur 3.10. För varje resurs erhålls datum som via historiken motsvarar sannolikheterna 5%, 25%, 50%, 75% och 95% för att de ska vara klara med sina uppgifter. Intervallet 25-75% är tydligare markerat. Bra estimerare utmärker sig genom smalare intervall.

Exemplet i figur 3.10 visar hur Milton Richie och Jane Curry kommer att kunna vara färdiga med sina uppgifter klart senare än de andra, på grund av låg tillgänglighet eller att en stor del av projektets arbete är fördelat till de två. Om det är möjligt så bör alltså några uppgifter omfördelas.



Figur 3.10: Sannolikhetsfördelning för slutdatum per resurs.

Selektiva anpassningar

Alla beräkningarna ovan kan i FogBugz anpassas genom att begränsa vilka uppgifter som räknas in efter vilken version (eng: release) de tillhör eller vilken prioritet de har. Till exempel kan man snabbt få en uppfattning om hur mycket slutdatumet skiljer om de lägst prioriterade uppgifterna inte görs klart.

Problem och avvikelser som tas hänsyn till i EBS

En grundidé med EBS berör händelsen att det i efterhand uppkommer buggar som är relaterade till en uppgift. Tanken är att tidsangivelserna ska göras för en korrekt utförd uppgift, inte en slarvig insats inför en deadline. Jämförelsen mellan estimat och resultat blir inte rättvisande eftersom uppgiften sedan krävde mer tid i efterhand och det är bland annat just sådant som gör att projekt går ur tidsramarna.

Hur ska sällsynta extremfall som samlas in hanteras? Lika väl som en uppgift kan försenas väldigt mycket kan en uppgift som tros ta mycket lång tid vara löst mycket snabbt. Eftersom dessa fall händer så sällan kommer de också att räknas in i Monte Carlo-simuleringen lika sällan, vilket gör att de ska låtas vara som de är. Men om uppenbara misstag upptäcks som är fullständigt osannolikt angivna så ska dessa tas bort, de fyller ingen roll på samma sätt som de andra extremfallen i att spegla verkligheten i historiken.

3.6 Lösningar och slutsatser

3.6.1 Det generella problemet

Eftersom det behövdes ett projekthanteringssystem är huvuddelen av lösningen framtagandet av Westimate i sig. Det blir ett förslag på hur Westbahrs arbete ska struktureras i ett system.

Den miljö som problemställningen uppstod i har beskrivits och problemdomänen har undersökts. Med denna analys som underlag har kravspecifikationen vidareutvecklats och lett till en abstrakt lösning i form av systemets design.

Några av de allmänna stödområden som har tagits upp i analysen är fysisk och logisk struktur, informationslagring, åtkomst, kommunikation och kognitiva hjälpmedel - för att nämna några. I alla datasystem som ska vara användbara måste dessa och många fler aspekter fungera och samverka. Det har även kort nämnts att det behövs ett flertal hjälpmedel för att driva projekt, såsom tidsstyrning. I framtagandet blir det många godtyckliga val som avgör detaljerna eftersom möjligheterna är oändliga. Det är väldigt svårt att få mätbara resultat och det finns ingen möjlighet i detta arbete att jämföra olika allmäna designval i systemet. Det hade isåfall varit om någon enskild detalj hade valts ut och resten hållits konstant, men någon uppenbar sådan detalj har inte ansetts finnas.

Att lösa alla de aspekter som krävs för att få ett framgångsrikt system för projekthantering är en stor konst och en mycket större uppgift än vad det handlar om i detta arbete. Det som kan göras för att komma vidare och konkretisera lösningen är att gå tillbaks till den generella problemställningen och börja med att återkoppla till problemet att arbetsuppgifter lätt tenderar att falla mellan stolarna.

Tilldelning av uppgifter

För att komma till rätta med ansvarsfrågan bör systemet uppmuntra att varje uppgift ska vara tilldelad någon. Det bör även gå att ange ett förslag på flera resurser så att det finns ett delat ansvar fram tills uppgiften tas om hand. För att täcka in fall när användare anser att de inte vet vem som ska tilldelas.

Det ska vara lätt att lägga in den minsta möjliga informationen som behövs för att registrera en uppgift i systemet, som senare kan kompletteras. Så att allt uppmuntras att föras in så fort det uppstår. Detta kan stödjas genom lättåtkomliga snabbformulär i gränssnittet samt konfigurerbar mottagning av e-post. Systemet behöver sedan uppmärksamma användaren på ej tilldelade uppgifter och uppgifter som är föreslagna till flera. Arbetsflöden och åtkomst av information i systemet har betydelse för detta. Återigen handlar det i stort om hur delar av systemet ska utformas. Samt att det behöver gå ut direktiv i organisationen om att alla uppgifter ska hanteras genom systemet.

Behandling av uppgifter

Det handlar till stor del om att inte göra systemet för komplicerat. Om för många av verklighetens detaljer förs med in i systemet tappas nog lätt effekten av de rätta förenklingarna, vilket gör systemet mindre användbart. Om för många obligatoriska formulärfält eller andra hinder och regler läggs till kommer de att ignoreras om det går och ifall det inte går kommer systemet att användas med ökat missnöje eller kanske till sist undvikas helt [3].

I de vyer där uppgifters information matas in och behandlas ska det i så många fall som möjligt erbjudas möjligheten att ställa in ganska flexibla värden på en mindre uppsättning fält. Och sedan låta systemets användare arbeta med uppgifter på ett sådant sätt att ens handlingar och val kommuniceras genom systemets händelseloggar, för att förtydliga vilken information som har förändrats eller tillkommit. Gränssnittet blir då en fri palett som användarna kommunicerar läget genom på de sätt som de själva får komma överens om. Sedan får systemet ge stöd i form av att exempelvis markera när en uppgift ej är tilldelad någon.

Tidsestimering och uppföljning

Av analysen som har gjorts framgår de enkla förutsättningar som krävs för att kunna börja arbeta vidare med metoder för tidsestimering och uppföljning. Systemet behöver kunna behandla uppgifter i projekt som utförs utav resurser och det behöver finnas tillvägagångsätt för att göra tidsuppskattningar och rapportera åtgången tid. Teoretiskt

blir förutsättningarna inte mer komplicerade än så.

För bra resultat behövs dock mycket mer, utöver speciella metoder för tidsestimering och uppföljning. Ett stödjande gränssnitt och regler för hur tidsdatan överförs och vad som ska gälla för undantagsfall som när en uppgift har varit tilldelade olika resurser eller när uppgiftens estimat har ändrats, lagts till sent, kanske tilldelats ett extremt värde jämfört med uppföljningen och så vidare. Frågan blir om det också jämnar ut sig i mängden som Spolsky hävdar [4] eller om något sådant fall behöver räknas om eller uteslutas innan de används som historiskt underlag.

Eftersom estimat och tidsrapporter alltid har en osäkerhetsgrad i sig, som bör utjämnas eller i alla fall blir närmast obetydlig i större mängder om insamlingen är konsekvent, är det allra viktigaste att angivelserna sker så enkelt som möjligt. Det finns ingen mening med att räkna enstaka minuter eller alltid tvinga fram tidsdata för precis varenda uppgift.

3.6.2 Det specifika problemet

Det behövs smarta åtgärder för att lyckas med målsättningen att öka tidsmedvetenheten och stödja en utveckling av uppskattningsförmågan inför arbetsinsatser. Om systemet bidrar med en struktur som leder till nya enade arbetsflöden, är det i sig ett stöd och ett första steg på vägen. Men för att nå längre behöver systemet också kunna hantera och ta fram någonting mer ur den information som förs in.

En önskan var att kunna bemästra tidsramar bättre genom att få besked om hur projekt fortlöper och prognoser på när de kan vara klara.

Det finns modeller för detta som har använts frekvent inom traditionell projekthantering för husbyggen och liknande arbeten. Gantt och PERT-scheman har samma infallsvinkel till projekt som ganska väldefinierade flöden som en gång sätts upp i paket enligt Work Breakdown Structure och sedan utförs rakt av. Men detta passar sällan bra in vid IT-projekt och särskilt inte när det som i det här fallet handlar om konsultarbete på begäran. Verktyg som Microsoft Project förenklar förvisso när projektplanen behöver revideras genom att kunna rendera om graferna. Men en sådan automatisk omfördelning har nackdelen att den ofta behöver justeras manuellt eftersom resurser kan placeras på delmoment som en annan resurs skulle kunna utföra mycket snabbare eller med högre kvalitet. En annan svaghet, särskilt med Gantt, är att de i samma graf inte hanterar flera förlopp med mindre skillnader, utan de är låsta till en enda plan.

Fördelen kan vara att det är den kritiska mängden arbete som identifieras om det görs helt rätt direkt. Graferna är också lätta att kommunicera idéer med, det finns en vana bland ledare på större företag av dem och Gantts sätt att presentera projektets status i block som utförs parallellt är en tilltalande modell.

Men för konsultarbete inom IT-sektorn blir det lätt för naivt att använda en projektledares egna mått utan någon kritik eller återkoppling. Det märks på den stora andel av projekt som misslyckas på grund av överraskande förseningar där inga åtgärder - som att ta in mer resurser, med kundens godkännande om det finns stöd för att det skulle rädda projektet, eller reduktion av krav - har gjorts i tillräckligt god tid.

Användning av estimat

Att definiera och tolka krav eller uppgifter beror så mycket på mänskliga faktorer och situationen. Det går givetvis inte att datalogiskt förutsäga framtiden, genom estimat som beräknas. Men med många olösliga problem brukar det vara möjligt att närma sig genom användning av approximationer, finns den möjligheten även i detta fallet?

Ett antagande som kan göras är att arbetsuppgifter och projekt bör slutföras med ungefär liknande förseningar som har uppstått förut. I enskilda fall sker det nog sällan exakt, men kanske i längden. Att det borde räta ut sig eller följa en röd tråd av förbättring som framträder allt tydligare med tiden och mer erfarenhet. Historiken kan fungera som ett approximativt mått på vilka oförutsägbara förseningar som i genomsnitt kan väntas.

En annan möjlighet som inte har använts är att klassificera uppgifter efter dess förväntade komplexitet, inblandade tekniker och andra egenskaper för att sedan jämföra mot tidigare uppgifter som har klassificerats på samma sätt. En stor del av detta är att komma fram till vilka egenskaper som är intressanta och riktlinjer för hur deras värden i sin tur sedan ska klassas för nya uppgifter. Det kan bli svårt att få ihop en användbar modell för detta.

Uppskattningar görs för att det finns ett mått av osäkerhet. Det enda sättet att kunna göra bättre estimat är egentligen genom att minska på osäkerheten. Vilket är svårt men som alla egentligen vet bör det göras genom att förtydliga krav, definiera avgränsningar och försöka dela upp stora projekt i mindre omfattande delar med självständiga resultat.

För att kunna göra bra estimat gäller det också att alltid eller aldrig separera ren arbetstid från distraktioner, inväntanden och yttre problem. Att arbeta i en stabil grupp som inte förändras mitt i arbetet. Att få feedback på tidigare estimat och att det alltid är eget arbete som estimeras. Det är inte projektledaren som ska estimera projektets resurser utifrån deadlines och dylikt. Det är där det ofta verkar gå snett, projektledaren borde som tidigare nämnts ha andra medel till hands istället. Som att minska på målen för projektets aktuella iteration. Det är inte hållbart att bygga upp en modell kring en person ur varje projekt.

Projektplaner och andra hjälpmedel blir bra först när de inblandade människorna estimerar sig själva bra. Tanken från början var att få kontroll över att projekt kan levereras i tid. Men vad menas egentligen med det? En sådan målsättning strider mot verkligheten i de projekt där det tillkommer nya krav, där kunden själv kan vara tillfälligt frånvarande eller inte ha avsatt resurser för att hantera överföringar av ny teknik eller extra arbete som följer med. Som konsultfirma går det då sällan att stoppa upp och säga nej till att hjälpa till utan det kommer med stor sannolikhet innebära ett förlängt projekt med fler uppgifter att lösa.

Evidensbaserade justeringar

Joel Spolskys idéer om EBS, evidensbaserade scheman som tas fram genom justeringar med historiska data som underlag, är tilltalande av flera anledningar.

Att basera modellen på allt känt arbete för stunden stämmer väl överens med hur större delen av projektens livscykel på Westbahr ser ut. Med många fristående uppgifter som tillkommer efter hand utan att placeras in i en traditionell projektplan. I enstaka fall kan det för tydlighetens skull markeras att det föreligger en sekvensiell ordning i hur delmoment ska utföras, men oftast är det irrelevant.

Det är en mycket intressant idé att presentera leveransdatum genom en sannolikhetsfördelning. Det blir tydligt att det måste göras en tolkning och att tidsuppskattningar i större sammanhang som projekt kan kommuniceras därefter. Så att det går att välja hur detaljerat mått på säkerhet som är lämplig från fall till fall. Det blir då viktigt att användaren förstår något om vad som ligger bakom diagrammen för att kunna dra slutsatser. Det är viktigt att statistiken inte upplevs som en sanning. Det handlar om osäkerhet och det kanske inte heller är så viktigt att det är helt matematiskt korrekta metoder som används för att omforma eller proportionera upp det visuellt för att underlätta tolkningar, så länge det blir tydligare.

Även att följa förändringen på hur optimistiskt, medel och pessimistiskt leveransdatum förändras allt eftersom uppgifter tillkommer eller avklaras är tilltalande för Westbahrs del. Ihop med de aktuella värdena kan det kommuniceras till kund att om de uppgifter som har tillkommit det senaste ska tas med i nästa leverans bör slutdatumet påverkas si och så.

Bedömningen har gjorts att EBS är bäst anpassad efter en miljö som Westbahr. För att tekniken ska fungera bra krävs det att miljön det används i överensstämmer med de grundläggande antaganden som har gjorts i modellen över hur arbete fördelas. Uppgifter måste vara avgränsade och tydligt fördelningsbara. De ska kunna fördelas till en enda resurs och detta innan uppgiften utförs. På arbetsplatser där uppgifter är svåra att fördela och många resurser ofta är delaktiga vilket gör utförandet mer flexibelt passar det inte lika bra.

EBS med modifikation

På grund av avgränsningar som har behövt ställas på detta arbetet, blir slutsatsen att ta fram en egen förenklad och modifierad variant av EBS.

Omräkningar genom resursers scheman och tillgänglig tid kommer inte att hinna lösas, istället för leveransdatum får uppskattad arbetstid i timmar duga som mått. Det är högst tveksamt om denna utvidgning kommer att implementeras senare.

Den konkreta lösningen ska inkludera insamling av data och presentation av statistik. Det visuella stöd som ska införas, inspirerat från EBS, är en graf med Monte Carlo-genererade sannolikhetsmått, för hur mycket arbetstid som bör krävas, för att slutföra en mängd estimerade uppgifter. Uppgifter som utförs av en eller ett par resurser som det finns tidigare historik för. Monte Carlo-simuleringens rundor ska fördelas ett sannolikhetsmått mellan 5 och 95%, enligt definitionen i EBS. För framtida rendering av den förändringsgraf, som används i EBS för att visa om projektet går mot ett avslut, ska det även vara möjligt att varje dygn lagra aktuella mått för utfallen närmast 5, 50 och 95%.

Hur övriga modifieringar exakt kommer att skilja sig från förlagan är okänt eftersom det saknas kännedom om hur många interna detaljer i EBS utförs.

Datainsamling

Den data som behöver samlas in, lagras och eventuellt behandlas eller underhållas består av estimat och tidsrapporter.

Det allra viktigaste är vilken data som samlas in för i efterhand kan statistiken givetvis genereras om med modifieringar om rådatan innehåller allt det som krävs. Inses det sedan att nya modeller inte får allt de behöver från strukturerna som samlats in är tidigare historik naturligtvis ej användbar i de fallen. Har man otur är de inte heller förenliga varför man tvingas välja att lagra redundans eller överge någon av modellerna [2].

Sedan är det viktigt att historisk data är så nära sanningen som möjligt. Det ska inte gissas, utan det bästa vore om det gick att arbeta med en sak i taget utan större avbrott utav andra uppgifter så att en klocka hade räckt för att hålla reda på tiden. Som vanligt är verkligheten inte så enkel.

Vilka resultat kan förväntas?

För att börja samla in data på estimeringar och dess verkliga åtgång i en organisation som inte har gjort det innan krävs ett stort engagemang från både projektledare och

deltagare för att komma till sin fulla rätt. Det krävs antagligen en kritisk massa som måste samlas in innan det blir vettigt. Om användningen är begränsad eller fläckvis är det svårt att få några bra resultat, det blir då även svårt att avgöra om statistikmodellerna är färdiga och realistiska. Om resultaten inte känns relevanta kan insamlingen givetvis upplevas som onödig, och att systemet lika gärna kunde vara enklare gjort. Uppnår man däremot den kritiska massan och modellerna ger bra resultat får man ett mycket kraftfullt och användbart system.

Det är svårt att förutse vilka resultat som kan väntas när systemet och stödet införs. Finns det någon mening med detta? Enligt en äldre artikel från Joel Spolsky år 2000 när han arbetade med att införa en enklare planeringsmodell, jämfört med EBS, skrev han "I've found that most programmers become very good schedulers with about one year of experience" [5]. Vilket mjukvarustöd som används är kanske inte det viktigaste bara det är anpassat till situationen och det bidrar till att förfina tankeprocesserna gällandes estimat.

4 Resultat

4.1 Design

Designen av systemet har utgått ifrån kravspecifikationen och de riktlinjer som fanns för systemets arkitekturella utformning. Förväntningen var att Westimate skulle komma i form av en webbapplikation. Det skulle utöver funktionaliteten som den specifika problemställningen belyser erbjuda flera av de funktioner som många bugtrackers / ticketsystem gör vid hantering av uppgifter och projekt. Några av de viktigaste som identifierades för Westimate var:

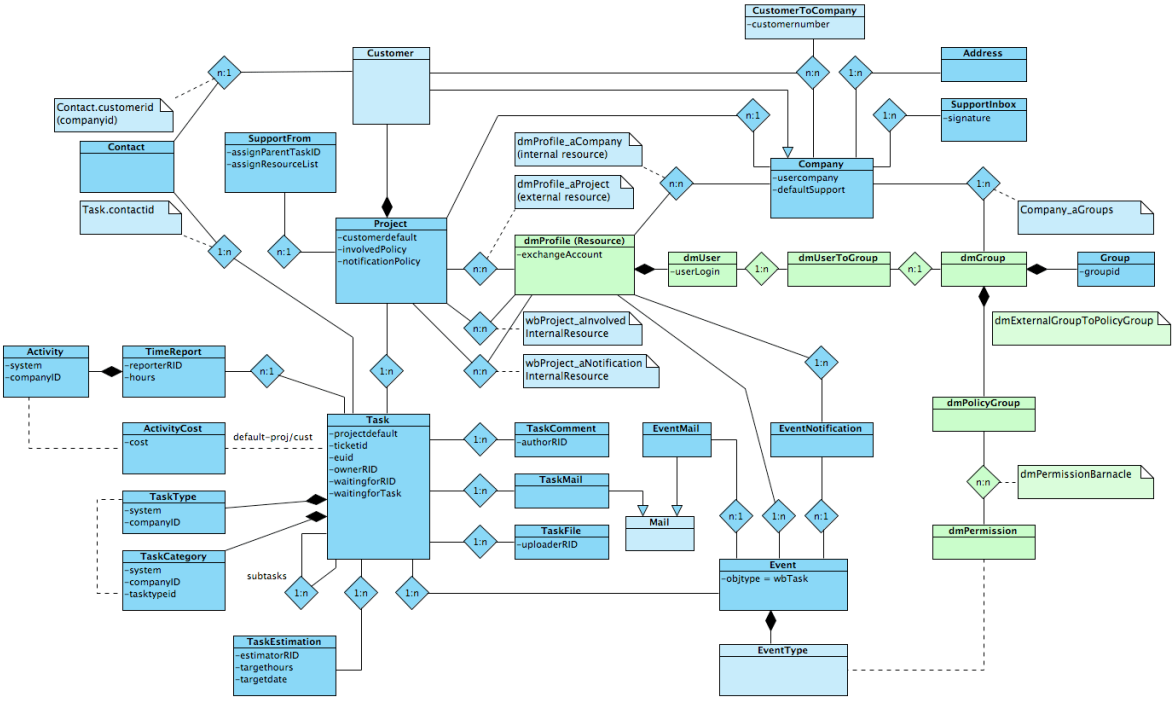
- Uppgiftsfördelning - flexibelt med möjlighet att föreslå flera resurser.
- Notifieringar och händelseloggning - både internt och via e-post.
- Hantering av supportärenden, eller vilken uppgift som helst, delvis via e-post.
- Utvalda kunder ska själva kunna lägga in och administrera ärenden.
- Filhantering - på så väl kund, projekt och uppgiftsnivå.
- Internkommunikation - genom en logg utav kommentarer.
- Överblickbarhet - exempelvis genom olika sorters urvals- och översiktsvyer.

Det fanns även många idéer inför framtiden som kunde vara bra att tas hänsyn till i designen:

- Möjlighet att styra vilken sorts statistik som tas fram för enskilda projekt.
- Återvinning av projekt - kunna utgå från ett projekt för att få ett skelett till ett nytt projekt.
- Möjligheten att dokumentera valfria fysiska eller virtuella objekt samt koppla dem till uppgifter. För att kunna organisera uppgifterna efter sådana objekt och för att kunna spåra samband eller enstaka källor till återkommande problem.
- Resursplanering och schemaläggning.
- Projektekonomi - följa upp lönsamhet via beräkningar på tidsåtgång och intäkter för olika aktiviteter.
- Hantering av olika releaser/versioner.
- Eskaleringsnivåer - exempelvis vid obehandlade eller oavklarade uppgifter.
- Händelsetriggers - som till exempel skickar meddelande eller fördelar vidare när ett datum har nåtts eller något mer avancerat villkor är uppfyllt.

4.2 Klass/domän-diagram

Utefter analysen och kravspecifikationen samt insikter som kommit under implementationen är systemets databas modellerad enligt diagrammet i figur 4.1.



Figur 4.1: Klass/domän-diagram.

De grönfärgade klasserna är datatyper som är fördefinierade i Farcry CMS. Resterande är specifika datatyper som har tagits fram för Westimate.

4.3 Viktiga designval

4.3.1 Företag, kunder, projekt och uppgifter

Företag

Det finns två typer av företag. Användarföretag har resurser och projekt som använder eller används genom Westimate. Övriga företag är i princip kunder till användarföretagen.

Kund

En kund skapas genom att en kundrelation bildas mellan två företag, ett användarföretag och ett kundföretag.

Projekt

Den struktur som ligger till grund för organiseringen av företag, kunder, projekt och uppgifter förutsätter att det för varje kund finns ett defaultprojekt. Ett projekt är ett sammanhang att samla arbete inom och för att inte behöva skilja på om arbete läggs in i ett projekt eller direkt mot en kund så används detta projekt, som skapas automatiskt per kund.

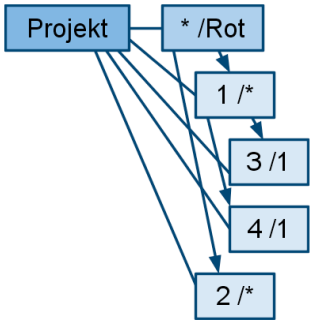
Alla projekt måste tilldelas ett användarföretag - som utför arbetet - och en kund. Ett användarföretag är per definition alltid en kund till sig själv, för att täcka in interna projekt. Således har varje användarföretag även alltid ett internt defaultprojekt som är det projekt som härstammar från relationen till sig självt.

Uppgift

På samma sätt som för relationen mellan kund och projekt finns det alltid en defaultuppgift som motsvarar ett projekts uppgiftsrot. Detta för att förenkla relationer från andra typer som kan beröra såväl uppgifter som projekt och i förlängningen, via projektets relation, även kunder. På så vis finns det ingen inbyggd begränsning i att till exempel filer eller kommentarer på sikt inte kan relateras direkt till ett projekt eller en kund om det skulle bli önskvärt.

Alla uppgifter som placeras i projektroten har en relation till defaultuppgiften - som egentligen upphör att vara en uppgift och istället verkar som en katalog för andra uppgifter. Även andra uppgifter fungerar så. När en uppgift placeras under en annan, antar den senare rollen som katalog fram tills att den eventuellt inte har några uppgifter under sig längre.

Denna trädstruktur förgrenar sig ned ifrån projektroten och samexisterar med att alla projektets uppgifter också har en enkel relation till projektet, se figur 4.2.



Figur 4.2: Projektstruktur.

4.3.2 Uppgiftsestimering

Varje estimering av en uppgift sparas separat, för att den kompletta historiken ska finnas lagrad. Enligt analysen räcker antagligen det senaste estimatet vid beräkningar på projektnivå, men det finns ingen större nackdel i att spara alla steg jämfört med att enbart lagra det senaste värdet i uppgiftstabellen. Konsekvensen är att det kan tillkomma logik vid all användning av estimat i databasfrågor och vid inhämtning av sammanställd data som berör en uppgift. Den största fördelen är att historiken kan användas för att följa upp utförandet av en enskild uppgift i detalj.

Lagringen sker i typen wbTaskEstimation, där ett estimat i arbetstimmar och/eller måldatum anges. Samt vilken resurs som lade estimatet. Ett tag i början fanns också möjligheten att uppskatta hur många procent av uppgiften som var utförd, men det togs bort då det knappt användes när systemet togs i bruk.

4.3.3 Tidsrapportering

Även tidsrapporter lagras var för sig i en egen tabell, wbTimeReport. Det behövs för att hantera fall som när olika resurser har haft en uppgift tilldelad sig och båda har rapporterat tid. För att kunna exportera tidsdata till andra applikationer blir det smidigast om varje tidspost lagras med en datumstämpel, vem som rapporterade och eventuellt vilken aktivitet som utfördes. Detta har att göra med de framtida idéerna gällandes projektekonomi som har modellerats in. Möjligheten att koppla timkostnader till aktiviteter i projekt eller för kunder ska eventuellt införas längre fram.

Vid hantering av totalt rapporterad tid per uppgift eller när rapporterad tid för en uppsättning uppgifter ska hämtas ut är det en nackdel att extra databaslogik behövs. För att optimera detta senare skulle sammanräkningen istället kunna ske när tidsrapporter läggs till, ändras eller tas bort och att detta värde lagras parallellt i uppgiftstabellen. Problemet blir då istället att se till att det hela tiden hålls synkat. Det samma gäller för uppgiftsestimeringarna. Men detta har inte införts ännu då laddningstiderna bedöms som godkända efter kraven som ställts.

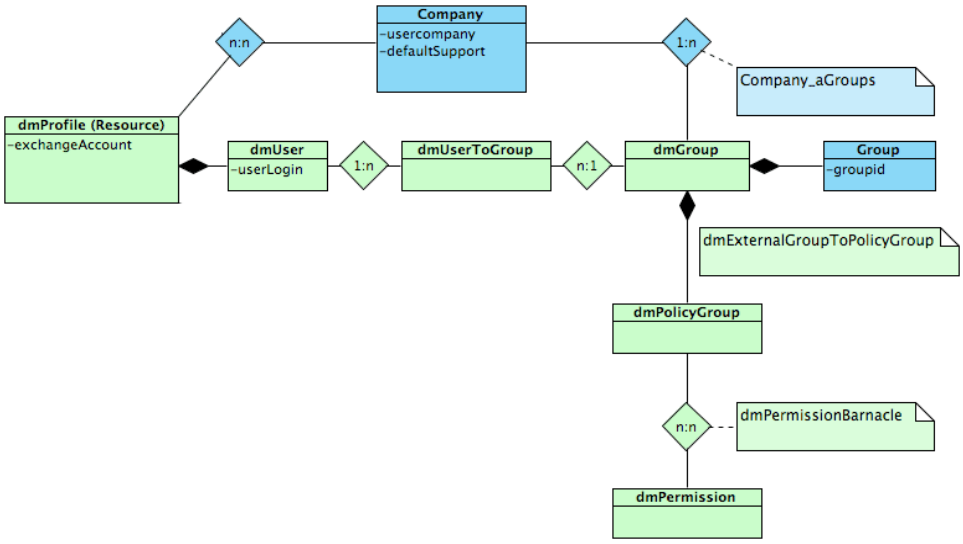
4.3.4 Typ/Kategori

Uppgifter kan definieras som en typ och kategoriseras. Uppgiftstyper kan också kategoriseras. Det innebär att en uppgift antingen kan vara klassad som en kategoriserad eller okategoriserad typ, kategoriserad men ej typad eller varken kategoriserad eller typad.

4.3.5 Användarplattform

Den plattform som utgör systemets ryggrad har byggts ovanpå Farcry CMS och dess användar/säkerhetsmodell. Hanteringen av profiler/användare, grupper och rättigheter går genom det API och den objektmodell med färdiga typer som ingår.

Detta val har gjorts för att undvika att hantera den grundläggande säkerheten själv, eftersom det är komplicerat och tidskrävande. Med en öppen lösning som använts och granskats av andra bör de flesta möjliga brister redan ha upptäckts och åtgärdats.



Figur 4.3: Användardelen av klass/domän-diagrammet.

Resurser

En resurs i Westimate är det samma som en person, det är den enda resurstypen som har identifierats för systemet.

Systemtypen dmProfile, som representerar en resurs i Westimate, har genom modifieringar i en ärvd applikationsspecifik definitionsfil utökats. Med fält för att hantera ett Microsoft Exchange-konto per användare samt ytterligare personlig data och inställningsval i Westimate. Resterande systemtyper är oförändrade. Ovanpå denna kärna har den övriga datamodellen med egna typer byggts till.

Typen dmUser används tätt förknippat med dmProfile. Den behandlar det basala, som användarnamn, lösenord och vilket så kallat User Directory som används för att lagra dessa uppgifter. Styrningen av rättigheter för en resurs går via kopplingen till användargrupp(er) från dmUser.

Relationstabeller till wbCompany och wbProject har implementerats genom varsin Array Table i dmProfile, de heter dmProfile_aCompany och dmProfile_aProject.

Intern resurs

En resurs som har kopplats till ett företag betecknas som intern resurs hos det företaget.

Extern resurs

En resurs som har kopplats till ett projekt hos ett företag där resursen inte är intern blir en extern resurs i det projektet. Detta syftar bara på den externa kopplingen som skapas genom en relation i dmProfile_aProject. Interna resurser kopplas till sina lokala projekt på ett annat sätt, genom en lista i projektet.

Rättigheter

Rättighetsmodellen i Farcry är baserad på policys och användargrupper.

En användare har tillgång till en skyddad del av systemet då den ingår i en användargrupp vars policy för den tillgången är godkänd. En policy beskriver en begränsning i helt vanliga ord. Det är upp till inblandade parter att avgöra hur den ska tolkas.

Företagstypen wbCompany har en Array Table (wbCompany_aGroup) för relationer till dmGroup för att kunna koppla en användargrupp till ett företag. Det finns även en wbGroup som enbart används för att ge användargrupperna ett obskyrt externt ID, för att säkert kunna koppla resurser till grupper fritt genom gränssnittet eftersom ID i dmGroup är ett löpnummer. Det går inte att utöka dmGroup som med dmProfile på grund av att säkerhetskärnans typer är slutgiltiga

För varje ny användargrupp skapas det även en policygrupp (dmPolicyGroup). De kan sedan mappas om så att flera användargrupper kan dela policygrupp. Valfria policys skapas och ställs in per policygrupp genom Farcrys API i applikationen eller Farcry Webtop - dess adminverktyg. I applikationen anges sedan en policy till checkPermission() som undersöker den autensierade användarens grupptillhörigheter för att avgöra om funktionen ska evalueras som sann eller falsk.

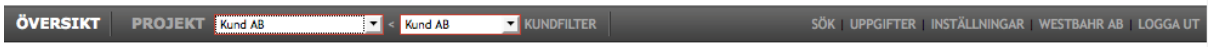
Några policys har inte implementerats fullt ut i Westimate utan denna delen har endast undersökts och testats. Beslut om lämpliga policys och annan begränsande strukturering har lämnats öppet.

4.4 Övergripande beskrivning av systemet

Denna del belyser Westimate från ett top-down perspektiv med en genomgång av gränssnittets huvudsakliga delar och deras inbördes samverkan.

4.4.1 Navigationsmeny

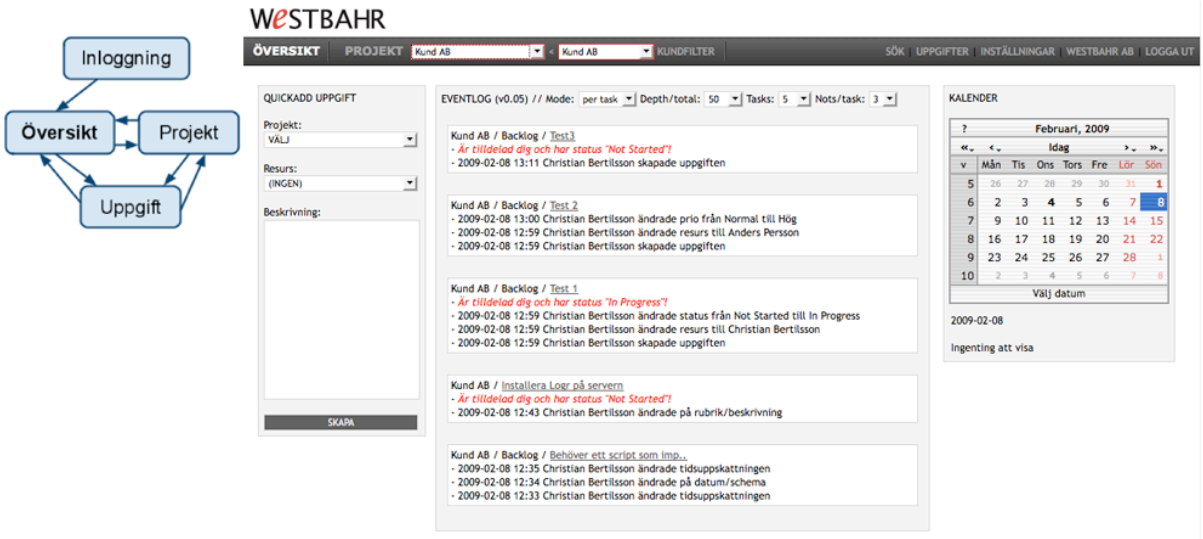
Högst upp finns en navigationsmeny, se figur 4.4, som från vänster består av en länk till översiktsvyn, en projektväljare inklusive kundfilter och ytterligare några länkar till vyer för sökningar, visning utav uppgifter som berör användaren samt användarens och systemets inställningar.



Figur 4.4: Navigationsmenyn.

Projektväljaren används för att navigera in till projektvyn för ett bestämt projekt. Eftersom antalet projekt lätt växer finns kundfiltret som ser till att det bara är den valda kundens projekt som visas och kan väljas. Kundfiltret lades till istället för att utesluta användarna från alla projekt de inte är involverade i, för att undvika att behöva administrera all projektåtkomst i början. Även om tillgången till projekt ska begränsas för de flesta resurser finns det roller som ska ha full tillgång. Resurser kommer dessutom att behöva deltaga i projekt tillfälligt, utan att de är involverade eller har uppgifter tilldelat sig sedan tidigare.

4.4.2 Översiktsvy



Figur 4.5: Översiktsvyn.

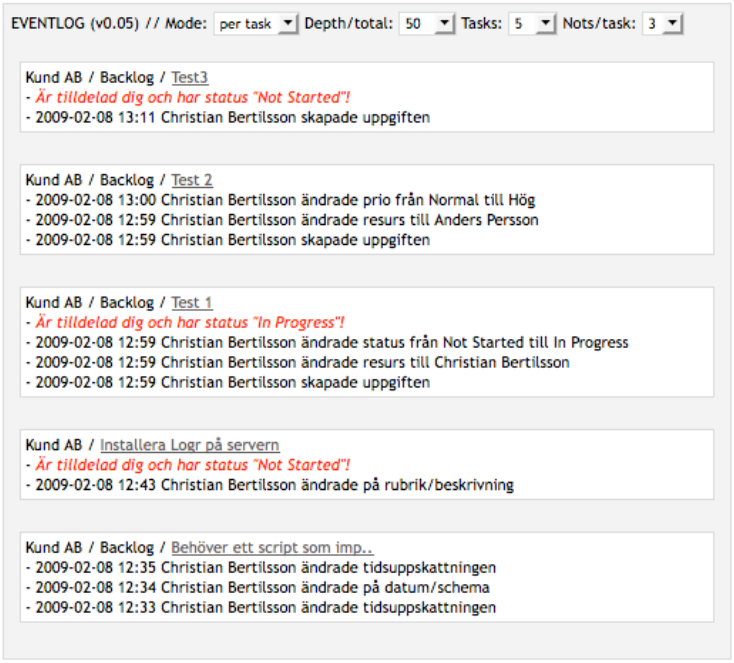
Arbetsytan i Westimate har utformats med en översiktsvy där meningen är att sammanställa sådant som användaren ska ha lättillgängligt för att få en snabb överblick över det totala läget i sitt arbete. De moduler som har placerats här inledningsvis i det syftet är en händelselogg och en kalender.

Händelselogg

Denna modul visar notifikationer för de senaste händelserna på uppgifter i projekt som användaren/resursen är utvald att bli notifierad på. Per projekt kan det väljas både vilka som notifieras och vilka som arbetar i projektet.

Notifikationerna kan visas grupperat per uppgift, som i figur 4.6, eller helt linjärt i en enda lista sorterat på tiden för händelserna. För att begränsa laddningstiden för översiktsvyn visar händelseloggen från början som mest fem uppgifter. För varje uppgift visas de tre senaste notifikationerna. Det görs först ett urval av de femtio senaste notifikationerna som grupperingen baseras på, om de alla skulle beröra samma uppgift så visas alltså bara den uppgiften. Därför finns möjligheten att ändra dessa parametrar genom rullgardinslistorna överst i modulen.

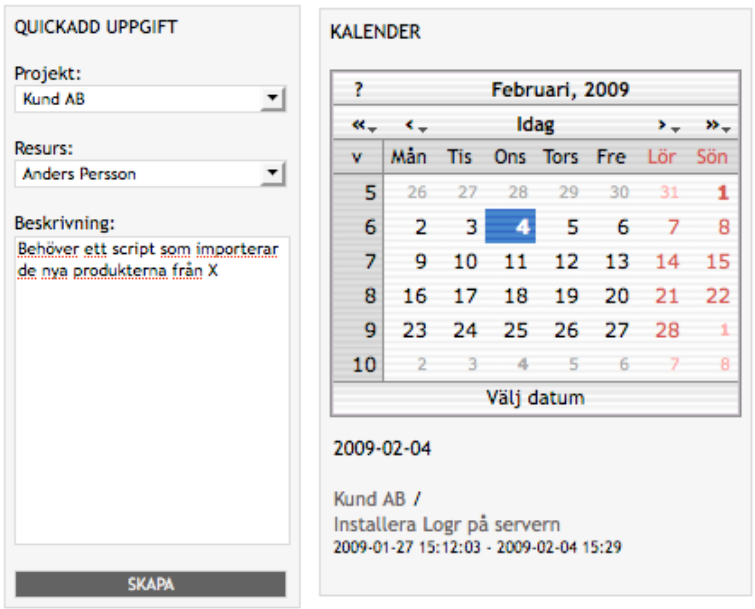
När notifikationerna visas i grupperat läge visas även meddelanden i rött, exempelvis när uppgifter är tilldelade eller föreslagna till användaren och uppgiften behöver behandlas. Det finns länkar från notifikationerna till vyerna för att visa uppgiften eller projektet den är placerad i.



Figur 4.6: Den övergripande händelseloggen.

Snabbformulär

Det går att enkelt föra in en ny uppgift i systemet till ett projekt genom ett snabbformulär, se figur 4.7. Tanken är att enbart den minsta information som behövs ska anges, för att uppmuntra att så mycket som möjligt förs in direkt och sköts genom Westimate. Till skillnad från projektväljaren i navigationsmenyn, som har sitt kundfilter, finns här bara projektvalet. Urvalet begränsas genom att det bara är de projekt användaren är involverad i som visas.



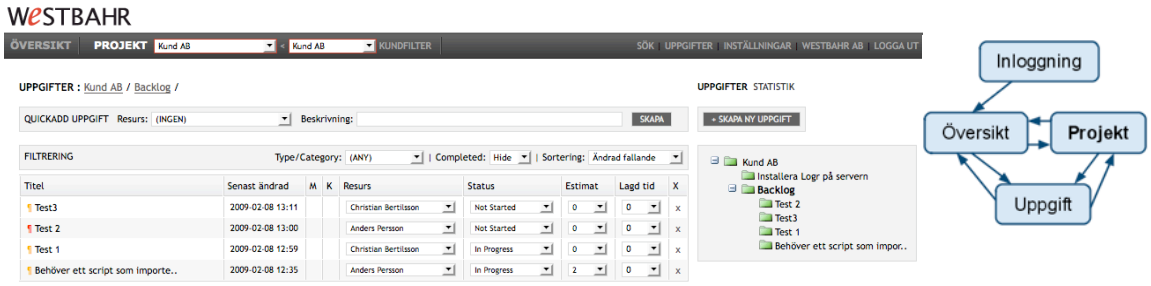
Figur 4.7: Snabbformulär respektive kalender.

Kalender

Kalendermodulen, se figur 4.7, vars grundfunktionalitet är utvecklad av Mihai Bazon och licensierad under GNU/GPL, används för att enkelt nå uppgifter där datumet har

betydelse. Uppgifterna kan schemaläggas eller få en deadline. Både start och slutdatum kan väljas att markeras för en uppgift i kalendern. Uppgifter med ett estimerat måldatum täcks också in. När ett datum markeras av användaren visas de uppgifter som är berörda på ovan nämnda sätt med en länk.

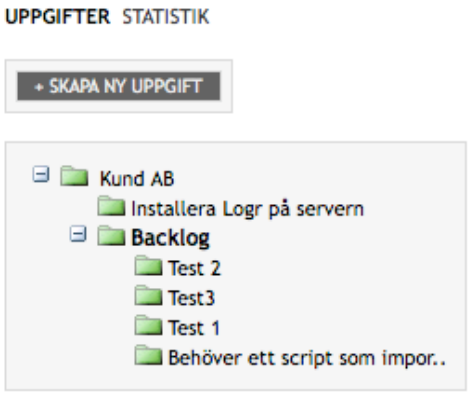
4.4.3 Projektvy



Figur 4.8: Projektvyn.

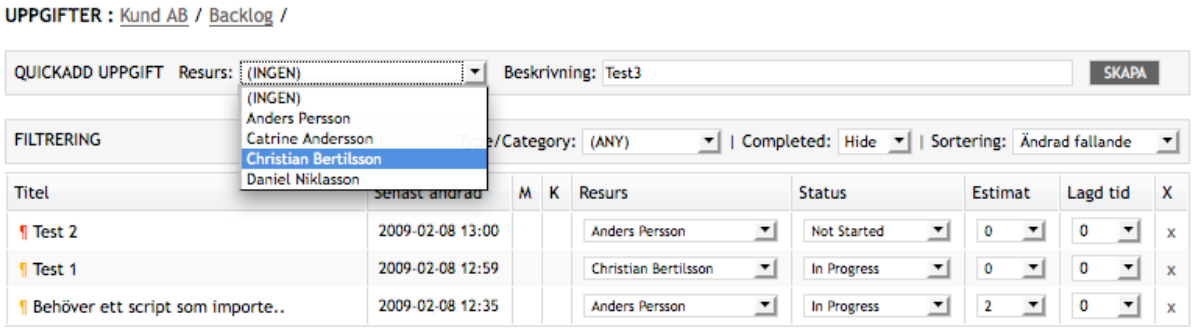
Även i projektvyn finns det ett snabbformulär för att enkelt föra in nya uppgifter. För att kunna ange alla detaljer finns möjligheten att gå till det kompletta formuläret via knappen "Skapa ny uppgift".

Ovanför knappen "Skapa ny uppgift" finns en meny för att välja vilken del av projektvyn som visas. Denna meny har inte utvecklats så mycket längre än att bestå av valet att visa projektets uppgifter eller statistik. Tänkta sektioner för att visa projektets filer och en enad händelselogg för projektet har ej implementeras ännu.



Figur 4.9: Meny och projektträd.

Projektets uppgifter och de uppgifter som verkar som kataloger visas i ett träd till höger i vyn. Det enda sättet att särskilja dem är genom att se om de har några uppgifter under sig eller inte. Detta indikeras genom en plus/minus-symbol framför, beroende på om katalogen visas öppen eller stängd. Det har inte prioriterats att införa olika ikoner för uppgifter och kataloger, men det är en god idé att göra.



Figur 4.10: Snabbformulär och en katalogs innehåll i en tabell.

Innehållet i den katalog som är vald i trädet visas i tabellen som täcker större delen av projektvyn, alltså när projektvyn visar uppgifter och inte statistiken eller någon annan vy av projektet.

Urvalet av uppgifter kan sorteras och filtreras. Filtret kan ta fler parametrar än "typ/ kategori" och "visa/dölj färdiga" som är de som för tillfället har valts att användas.

Varje rad i tabellen består av det som har bedömts vara viktigast att se och snabbt kunna ändra för uppgifterna.

Att utforma de databasfrågor, som används för att sammanställa informationen som visas i tabellen har varit en problemfylld process. Att lagra redundans i uppgiftstabellen för de data som helt hanteras genom andra tabeller eller som behöver slås upp, eftersom enbart en referens lagras i uppgiftstabellen, hade underlättat avsevärt för laddningstiderna. Men lett till andra problem. Särskilt när de objekt som använts för att aggregera de statiska värden, som isåfall hade kunnat lagras i uppgiftstabellen, ändras eller raderas. Detta gäller bland annat för estimat och tidsrapporter. För andra egenskaper, som den utförande resursen skulle både namn och referens kunna lagras istället för bara referensen, men ändras då namnet senare behöver man se till att uppgiftstabellen synkroniseras. Om detta inte görs leder det till fel i redundansen, som kan vara svåra att upptäcka.

Den väg som har valts är att tills vidare låta bli att införa redundansen. Istället har databasfrågorna delats upp på ett sätt som löser problemet och ger acceptabla laddningstider, så länge det inte finns flera tiotals uppgifter i en katalog. Vid 20-25 uppgifter passeras den specificerade gränsen och användaren får då helt enkelt vänta lite extra. Uppdelningen sker genom att de komplexa kolumnerna i visningstabellen först hämtas var för sig för alla uppgifterna i katalogen, summerat tar de då inte alls lika mycket tid i anspråk, som om allt hade gjorts i ett enda steg. Sedan sammanställs raderna utifrån de olika databassvaren med logik i Coldfusion på servern.

Rullgardinslistorna som finns i flera av kolumnerna används för att snabbändra värden. När sidan laddas hämtas de möjliga valen en gång per kolumn och det aktuella värdet för alla uppgifterna i en enda databasfråga per kolumn. För att sätta det valda värdet anropas samma funktion som används när något av dessa värden ändras av användaren och ska sparas, vilket leder till att rullgardinslistan laddas om med det nya värdet valt. Fast istället för att köra en databasfråga per uppgift skickas all data in ifrån projektvyn vilket innebär att frågan bara behöver köras en gång per kolumn vid sidladdningen.

Projektvyns statistikdel

I figur 4.12 visas ett exempel på hur det kan se ut när projektvyns statistikdel visas. Denna del har ej hunnits implementeras färdigt lika långt som det var planerat, fokus har helt och hållet legat på den svåraste delen - att ta fram en variant av EBS.

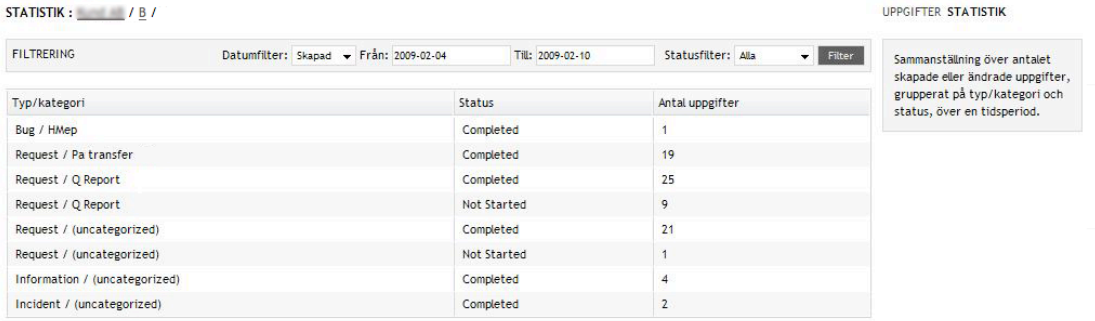
Således har inte grafen som visar hur ett optimistiskt, ett medel och ett pessimistiskt mått förändras över tid införts. Dels på grund av tidsbrist men även att det blir inkonsekventa resultat om algoritmen förändras, vilket den har gjort in i det sista. Det är lika bra att vänta med den grafen tills algoritmen har utvärderats och stabiliserats. Att lagra och visa historiken är inte någon svårare uppgift att lösa.

Inte heller den enklare "föregångaren" till EBS-grafen, processdiagrammet (eng: burndown chart), har hunnit implementeras tillfredsställande, så den har uteslutits att presenteras här. Likaså möjligheten att se hur många uppgifter som har varit obehandlade i veckor eller månader och annan mer grundläggande statistik som kravspecifikationen efterfrågade.

Det som däremot finns är:

- TCS (Type, Category, Status) som är ett rapporteringsverktyg för att kunna få ut statistik över antalet uppgifter som har skapats eller ändrats under en

tidsperiod, se figur 4.11. Denna del förklaras ej ingående, men notera att den inte har något att göra med grafen som visas i figur 4.12.


STATISTIK : [redacted] / B /

UPPGIFTER STATISTIK

FILTRERING Datumfilter: Skapad Från: 2009-02-04 Till: 2009-02-10 Statusfilter: Alla Filter

Sammanställning över antalet skapade eller ändrade uppgifter, grupperat på typ/kategori och status, över en tidsperiod.

Typ/kategori	Status	Antal uppgifter
Bug / HMep	Completed	1
Request / Pa transfer	Completed	19
Request / Q Report	Completed	25
Request / Q Report	Not Started	9
Request / (uncategorized)	Completed	21
Request / (uncategorized)	Not Started	1
Information / (uncategorized)	Completed	4
Incident / (uncategorized)	Completed	2

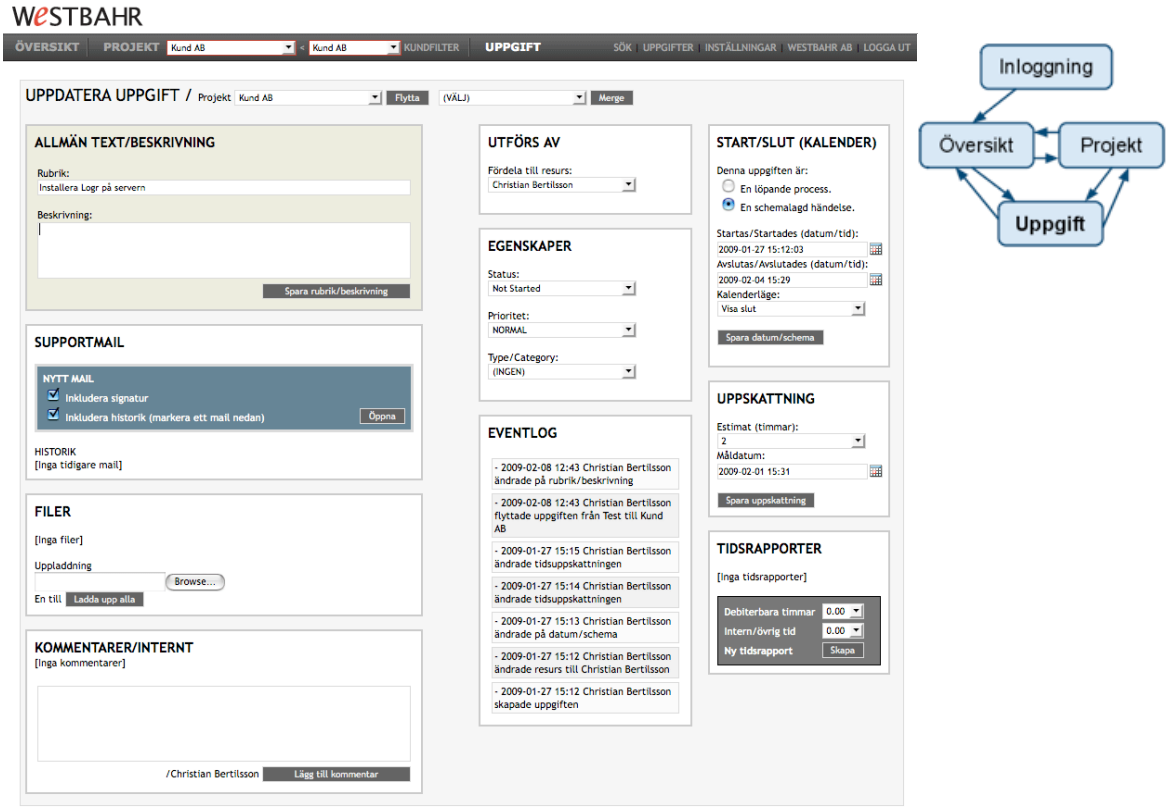
Figur 4.11: En TCS-rapport.

- EBS-inspirerad graf, se figur 4.12, som är det huvudsakliga stödet för användaren att handskas med estimat och projekts tidsramar. Hur den har implementerats beskrivs i ett senare avsnitt.

Figur 4.12: Projektstatistik genom TCS och en EBS-inspirerad graf.

4.4.4 Uppgiftsvy

Denna vy används för att se och behandla en uppgift i sin helhet. Det finns separata sektioner för det som kan göras med uppgiften och den information som kan föras in. Gränssnittet ser ut enligt följande:



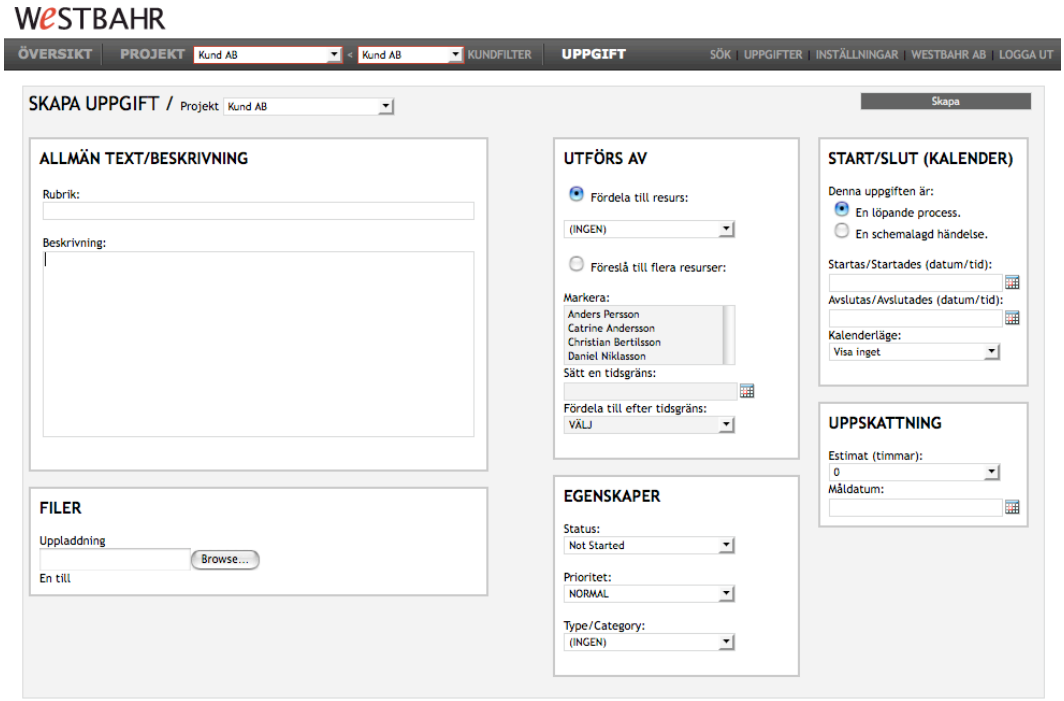
Figur 4.13: Uppgiftsvyn i redigeringsläge.

Illustrationen är bara ett exempel, många av sektionerna saknar här datainnehåll och ger inte hela bilden av de små finesser som framträder vid olika händelser.

Kravet att vyn ska vara uppdelad i flikar har ej mötts ännu. Vyns kompakta utförande är för tillfället mer användarvänlig för programmerare och tekniker än vad den är för kunder, säljare och andra typer av användare. I övrigt är de funktionella kraven för uppgiftshanteringen uppfyllda, och mer därtill för ett par av sektionerna som har utvecklats vidare.

I likhet med projektvyn kan uppgiftens tidsuppskattning föras in och redigeras. Här finns även möjligheten att ange ett måldatum som uppmärksammas i kalendern i översiktsvyn, användningsområdet för måldatumet är endast detta. Tidsrapporter kan också läggas in och redigeras i större detalj än från projektvyn. Tanken är att det i de flesta fall inte ska behövas navigeras runt bland uppgifter för att estimera och tidsrapportera i sin enklaste form, men att de detaljerade formulären finns här när måldatum är intressant att ange eller när särskilda tidsrapporter behöver mer information - det ska framöver gå att ange en aktivitet också.

När en ny uppgift skapas genom uppgiftsvyn är inte alla sektionerna tillgängliga. Och istället för att välja en enda utförande resurs kan ett förslag skickas ut. Vilket innebär att flera markeras som föreslagna och att de sedan får fördela uppgiften mellan sig - så att en uppgift inte är ansvarslös och tappas bort när det är oklart vem som ska utföra den.



Figur 4.14: Uppgiftsvyn vid skapande av ny uppgift.

4.5 Implementation av några utvalda koncept

4.5.1 E-postmottagning

Ett sätt att föra in uppgifter i systemet, vid sidan av snabbformulären och uppgiftsvyn, är att skicka e-post. En så kallad supportinbox, som motsvarar ett konto på en mailserver, kan skapas i inställningarna för användarföretaget. Dessa konton används för att ta emot e-post, som plockas in via ett skript som körs varannan minut i Westimate.

Varje supportinbox kan kopplas till en katalog i ett projekt, där inkommande e-post för nya ärenden skapas som uppgifter. För de supportinboxar där ingen katalog har angetts kommer mottagna ärenden att placeras i ett övergripande supportprojekt, vilket kan väljas i användarföretagets inställningar.

Den e-post som inkommer till alla supportinboxar går igenom ett filter. Filtret matchar e-postens avsändaradress mot en lista av adresser eller en regel, i form av ett reguljärt uttryck, som exempelvis kan motsvara en domän eller ett företagsnamn. En matchning är global för användarföretagets alla inboxar. Inkommande e-post testas mot alla inlagda matchningar i en godtycklig ordning tills en träff sker. Om avsändaradressen finns i listan eller uppfyller regeln för en matchning placeras det nya ärendet som en uppgift i matchningens utvalda katalog.

När e-post sänds ut från Westimate, som svar på inkomna ärrenden eller på eget initiativ av användare, tilldelas ämnesraden och uppgiften en sträng i formatet "[#123ABC]". Strängen är ett så kallat ticket-ID som används för att identifiera en konversation. När e-post med ett ticket-ID i ämnesraden inkommer till systemet, genom en supportinbox, och en uppgift med samma ticket-ID finns i systemet placeras meddelandet i den uppgiftens e-postkonversation direkt. Det sker utan att matchningsfiltret eller den mottagande supportinboxens kataloginställning används.

4.5.2 Microsoft Exchange-integration

När en uppgift sparas så försöker Westimate exportera den till Exchange, om användaren har valt detta i sina inställningar.

För att en uppgift ska exporteras ut som en Exchange Task till någons konto krävs det att:

- Den aktuella uppgiften har tilldelats en resurs
- Denna resursen har angett uppgifterna om sitt Exchangekonto i sina inställningar.

I uppgiften sparas den unika ID-sträng som objektet tilldelas i Exchange, för att kunna uppdatera samma objekt senare.

4.6 Implementation av en EBS-inspirerad algoritm

Här följer en detaljerad beskrivning av den algoritm som är det särskilda stöd i systemet som ämnar att erbjuda ett förslag på hur systemet kan handskas med estimeringar för att hjälpa användaren i sitt arbete med osäkra tidsramar. I appendix B finns den viktigaste källkoden sammanställd.

4.6.1 Algoritm

A. Inhämtning av grundläggande data

1. För varje involverad resurs i projektet hämtas r/e-faktorer (rapporterad/ estimerad tid) för de 100 senaste uppgifterna som har estimerats, tidsrapporterats, avslutats och varit tilldelad resursen. Om en uppgift tog 2x så lång tid som estimerat så är r/e-faktorn 2.
2. För varje uppgift i projektet ***** som är estimerad och ej avslutad hämtas hittills rapporterad tid och det senaste estimatet.

***** kan även vara en katalog

B. Monte Carlo-simulering

Följande sker för varje uppgift som hämtas in i A.2:

1. En resurs väljs. Om uppgiften är tilldelad en resurs och denna resurs är involverad i projektet är valet klart och inget mer behöver göras ******.
2. En r/e-faktor väljs ut slumpmässigt från den valda resursens historik.
3. Uppgiftens estimat justeras genom att multipliceras med den valda r/e-faktorn *******.

De värden som fås i beräkningen ovan summeras per runda för alla uppgifter. Simuleringen upprepas n gånger och varje rundas totala summa tilldelas ett sannolikhetsmått av 1/n %.

****** Är uppgiften tilldelad en resurs som ej är involverad i projektet behöver samma sak som i A.1 göras för den resursen (A.1 skulle nog kunna optimeras genom att det alltid sker här istället, en gång per resurs första gången resursen påträffas, så att resurser som ej är berörda av någon uppgift inte laddas in).

Om uppgiften inte är tilldelad någon resurs är chansen stor att den antagligen kommer att tilldelas någon snart och att det är stor chans att den kommer att tilldelas någon som utför mycket arbete i projektet. Därför simuleras en fördelning av sådana fristående uppgifter efter hur de fördelade uppgifterna är distribuerade bland projektets involverade resurser. Även andelen ej tilldelade uppgifter räknas in i fördelningssimuleringen och motsvaras av en helt slumpmässig fördelning, för att undvika fallet att det finns en betydande mängd ofördelade uppgifter och en mindre mängd fördelade vars enstaka resurs(er) då hade räknats som utförare av alla uppgifterna.

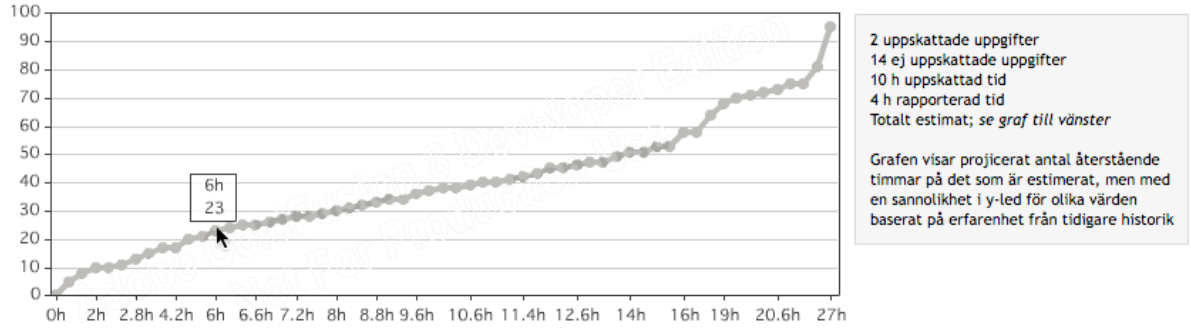
*** Detta bör också anpassas efter hittills rapporterad tid. Är ett estimat angett i 10 timmar och 8 timmar redan har rapporterats på uppgiften, ska det justerade estimatet inte kunna gå under 8 timmar. Observera att de rapporterade timmarna inte ska räknas in i summeringen, eftersom det är förväntad kvarvarande tid som mäts.

Det är inte lika självklart vad som ska gälla när det omvända sker, att mer tid än estimatet redan har rapporterats.

Den tillfälliga lösning som har valts för båda fallen är att addera den absoluta skillnaden mellan estimerad och rapporterad tid multiplicerat med r/e-faktorn. Det antas att estimatet borde ha redigerats om det uppenbarligen inte alls stämmer längre och att om den rapporterade tiden då har närmat sig eller passerat estimatet så bör uppgiften vara nästintill klar - och osäkerheten ligger då i skillnaden mellan värdena istället för bara i estimatet.

C. Visualisering

I Coldfusion ingår <cfchart> för att plotta grafer. <cfchart> har några begränsningar i layout och finesser men har bedömts vara fullt tillräcklig att använda. Taggen genererar en Adobe Flash-fil, som med de valda parametrarna visar punkter på två axlar genom enkel grafik.



Figur 4.15: Projektprognos genom en EBS-inspirerad graf.

I illustrationen ovan visas ett enkelt exempel med ganska dåliga historiska data som underlag. Simuleringen gav ~23% chans att estimatet för de 10 timmarna på de två estimerade uppgifterna ska hålla, att 4 timmar redan har rapporterats in ger att det är 6 osäkra timmar kvar.

Den liggande x-axeln motsvarar antalet timmar som de estimerade uppgifterna förväntas ta att utföra. Detta är de summeringar som fås ur Monte Carlo-simuleringens rundor. Dessa behöver enbart sorteras och plottas i ordning. Y-axeln med sannolikheter från 0 till 100% behöver dock motiveras lite närmare.

Eftersom simuleringen utförs n gånger kan varje runda tilldelas en sannolikhet på 1/n % *inom* dessa rundorna. Simuleringen ger ett öppet intervall eftersom det inte går att

avgöra hur de fiktiva rundorna n+1, n+2 till n+k hade givit summeringar som är lägre eller högre än det lägsta respektive det högsta värdet som simulerades.

Antagandet som görs här är att det kan finnas upp till ungefär 5% i vardera ända som ej täcks in av Monte Carlo-simuleringen. Joel Spolskys metod gör samma antagande för n=60 medan det är n=100 rundor som körs i Westimate så antagandet bör vara generöst nog.

Y-axeln baseras också på resultaten från simuleringens rundor, men det är skillnaden mellan rundorna och hur många summeringar som erhålls med liknande resultat som avgör höjden mellan punkterna och således grafens karaktär. Y-axeln ska alltid röra sig från 0 via 5% till 95%. 0%-värden plottas fram till den lägsta summeringen, som tilldelas sannolikhetsmåttet 5%, och resterande värden normaliseras därför enligt följande:

1) Sänks med värdet av den lägsta summeringen
2) Multipliceras med en faktor: (95 - 5) / (högsta - lägsta summeringen)
3) Ökas med 5

4.6.2 Reservationer

Notera igen att det är spekulativa sannolikhetsmått som behandlas, inte objektivt angivna sannolikheter. Det är också helt i sin ordning eftersom resultaten baseras på estimat.

Jämfört med Joel Spolskys EBS-algoritm görs ingen omräkning från arbetstimmar till datum beroende på scheman, mått i arbetstimmar får duga. EBS räknar även ut individuella scheman och anpassar slutdatumet efter den resurs som ser ut att vara sist med att bli klar med sitt arbete. Något sådant är inte heller behandlat här.

Uppgifter anges inte som beroende av varandra här eller i EBS på det viset som i traditionell projektplanering med vattenfallsliknande modeller. Inom projekten som berörs kan och bör uppgifter i mycket hög grad kunna utföras och stå på egen hand, det är sällan som det sker enligt större faser där en fas leder till en annan.

Specialfall som när en uppgift byter ägare eller olika personer har estimerat behandlas genom att den senaste ägaren alltid är ansvarig för vad som sker med uppgiften och får justera estimat därefter om det anses behövas. Att logga när en uppgifts ägare byts ut och räkna med segment utav uppgiftens utförandeperiod hade höjt komplexiteten på algoritmen till en orimlig nivå som ändå i de flesta fall inte ger några större genomslag för simuleringen i sin helhet.

5 Slutsats

I analysens avslutande avsnitt har flertalet av de viktigaste slutsatserna som låg till grund för implementationen presenterats. Här sammanställs några avslutande ord om arbetets resultat och framtid.

Det framtagna systemet uppfyller majoriteten av de krav som ställts. Systemet har tagits i drift av Westbahr AB och används även i mindre utsträckning av ett par närstående företag. Många utvidgningar och förbättringar finns kvar att göra, men det är helt i sin ordning eftersom det går väl ihop med den adaptiva utvecklingsmetoden som använts.

Algoritmen behöver utvärderas i avseende på hur bra stöd den ger. För detta krävs ett större underlag av historiska data och resultat från flera genomförda projekt. Algoritmen kommer att justeras och utvecklas vidare vid behov efter att en sådan utvärdering har gjorts.

Det behöver utredas närmare om sättet som ofördelade uppgifter tilldelas en resurs i simuleringen fungerar bra, och exakt hur rapporterad tid ska påverka omräkningen till osäker tid för lagda estimat.

En ny idé som skulle kunna undersökas är att när ett tillräckligt underlag finns i första hand använda historiska data för resursens uppgifter ur samma projekt som beräknas Istället för att rakt av använda resursernas globala historisk från alla projekt. Med tanke på att karaktären på uppgifter inom ett projekt säkerligen kan anta ett par mönster.

Att särskilja på hela projektet Westimate och den kärna som är intressant för examensarbetet - uppgiftshanteringen och estimeringsstödet - har varit omöjligt. Ambitionsnivån har varit för hög. Totalt sett har det mot slutet upplevts som att tämja ett olösligt eller åtminstone oöverskådligt problem att lyckas sammanställa motiveringarna bakom hur systemets många berörda delar är uformade.

Det är ett svårt och öppet område som har behandlats i detta arbete. Det har sällan funnits några uppenbara rätt eller fel att använda sig av, teorier och resonemang har bedömts och valts ut som lämpliga för just denna situation. En svaghet med arbetet är att många beslut har grundats på en informell diskussion med vaga termer hämtade ur projekthanteringsområdet. Samtidigt är det i den miljön som problemet har undersökts, vilket gör det angreppssättet svårt att undvika.

Förhoppningen är att detta arbete har utförts på ett lämpligt sätt och att det uppställda syftet, om Westimates positiva konsekvenser på sikt, kommer att nås med all tydlighet när systemet har varit i användning ett tag till och utvecklats vidare.

Litteraturförteckning och referenser

[1] Steve McConnell, Software Estimation: Demystifying the Black Art, Microsoft Press, 2006.
[2] Camilla Lindström & Anna Lindvall, An Estimation Improvement Method Aiming for Successful Deliveries of Software Projects, Chalmers Tekniska Högskola, 2006.
[3] Joel Spolsky, More Joel On Software, Springer-Verlag New York, 2008.
[4] Joel Spolsky, Evidence Based Scheduling, från bloggen "Joel on Software", http://www.joelonsoftware.com/items/2007/10/26.html, hämtad i december 2007.
[5] Joel Spolsky, Joel On Software, Springer-Verlag New York, 2004.
[6] James A. Highsmith III, Adaptive Software Development: A Collaborative Approach to Managing Complex Systems, Dorset House Publishing, 2000.
[7] Project Management Institute, A Guide to the Project Management Body of Knowledge - 4th edition, 2008
[8] Ottersten & Berndtsson, Användbarhet i praktiken, Studentlitteratur AB, 2002.
[9] Microsoft Project http://www.microsoft.com/Project/
[10] Coldfusion, http://www.adobe.com/products/coldfusion/
[11] Robert Marks, A Brief History of ColdFusion, http://www.tophosts.com/articles/?3016.html, hämtad 2008.
[12] R. Lerdorf, Announcing the Personal Home Page Tools (PHP Tools) version 1.0, http://groups.google.com/group/comp.infosystems.www.authoring.cgi/msg/cc7d43454d64d133?pli=1, hämtad 2008.
[13] Farcry CMS, http://www.farcrycore.org
[14] B. Boiko, Content management bible, Wiley Publishing, 2004
[15] Microsoft Exchange, http://www.microsoft.com/exchange
[16] ajaxCFC, http://ajaxcfc.riaforge.org
[17] JSDragDropTree, http://www.dhtmlgoodies.com
[18] JSCalendar, http://www.dynarch.com/projects/calendar
[19] K. Schwaber, Scrum Development Process, 1995.
[20] H. Kniberg, Scrum and XP from the Trenches, C4Media Inc, 2007.
[21] Projektplatsen.se / Projectplace, http://www.projectplace.se
[22] Projektbarometern http://www.projectplace.se/Kunskapsplatsen/Artiklar-Rapporter/Rapporter/ProjektBarometern/
[23] Y. Edenholm, Allt fler it-projekt misslyckas, http://www.nyteknik.se/nyheter/it_telekom/allmant/article43394.ece, 2007
[24] TicTac Mobile, http://www.tictacmobile.com
[25] FogBugz, http://www.fogcreek.com/Fogbugz
[26] Devshop, http://www.devshop.com
[27] Devshop 2.0 lanseras, www.prlog.org/10097269, hämtad 2008
[28] C. Bentley, PRINCE2: A Practical Handbook, Butterworth-Heinemann, 2001
[29] P. Kruchten, The rational unified process: an introduction, Addison-Wesley Longman, 2000
[30] L.R. Shaffer & J.B. Ritter & W.L. Meyer, The critical-path method, McGraw-Hill Inc, 1965
[31] R.C. Tausworthe, The work breakdown structure in software project management, Journal of Systems and Software, 1980
[32] H.L. Gantt, Work, Wages and Profit, The Engineering Magazine, 1910
[33] Daniel D. Roman, The PERT system, Journal of the Academy of Management, 1962
[34] FogCreek, Top Ten Tips for Successful Bug Tracking, http://www.fogcreek.com/fogbugz/docs/30/TopTenTipsforSuccessfulBu.html
[35] World Wide Web Consortium - Web Standards, http://www.w3.org/
[36] MySQL, http://www.mysql.com
[37] Coldfusions historik sammanställd, http://en.wikipedia.org/wiki/ColdFusion
[38] D. Hubbard, How to Measure Anything: Finding the Value of Intangibles in Business, John Wiley & Sons, 2007
[39] D. Hubbard, The Failure of Risk Management: Why It's Broken and How to Fix It, John Wiley & Sons, 2009
[40] J. Woller, The Basics of Monte Carlo Simulations, University of Nebraska-Lincoln, 1996

Appendix A: Kravspecifikation

Westimate
Kravspecifikation
Version <1.0>

A.1. Inledning

A.1.1 Bakgrund

Westbahr är ett utvecklingsbolag med fokus på att ta fram innovativa lösningar och att hjälpa företag att effektivisera sin IT-miljö. Företaget vill förbättra strukturen och de egna rutinerna. Sedan starten har svårigheterna med att estimera tidsåtgång i projekt och uppföljningen av den verkliga åtgången blivit allt tydligare.

Det förekommer även att arbetsuppgifter hamnar mellan stolarna och att den interna kommunikationen gällandes vad som sker i projekten, vad som behöver göras och hur det har gjorts tidigare kan effektiviseras avsevärt. Dokumentation av uppgifter och tillvägagångssätt inom projekten ska kunna återanvändas samt vara tillgängliga för anställda.

Ett smidigt supportflöde och mindre stress över om något viktigt kan ha glömts av är också av stort intresse.

Det finns ett behov av att ta fram ett verksamhetsstöd som löser detta och ger företaget ett tydligt grepp om sina kunder och projekt.

Många koncept inom tidsplanering och projekthantering finns redan framtagna, fokus i detta projekt ligger på att designa bra struktur och hitta smarta sätt att täcka in många av de undantagsliknande situationer som kan komma att uppstå. En mångsidig grundstruktur med datainsamling vid många händelser ger goda möjligheter att presentera flera olika modeller gällandes statistik för uppgifter, projekt och anställda resurser.

Både struktur och statistikmässigt är målet att stödet ska vara anpassat för att bistå situationen på Westbahr i första hand. En analys med en förstudiedel har genomförts varav dess resonemang och slutsatser ligger till grund för systemets utformning.

A.1.2 Syfte

Syftet med denna kravspecifikation är att ge en översikt av den önskade funktionaliteten samt definiera objekt och koncept. Detta görs med en genomgång av systemets mest centrala delar och deras roller. Dokumentet går även in på allmänna krav som baseras på miljön systemet ska placeras i och vilka begränsningar eller angränsade krav det medför.

A.1.3 Definitioner, förklaringar och förkortningar

A.D.1 Westimate
Namnet på systemet detta dokument beskriver.

A.D.2 Uppgift
Arbetsuppgift, motsvarande engelskans task. Alltså inte synonymen till information.

A.D.3 Katalog
När en uppgift tilldelas andra uppgifter under sig blir den katalog för dessa, enligt en parent/child-relation.

A.D.4 Uppgiftstyp
Uppgifter kan genom typ särskiljas som exempelvis arbetsuppgift, bugg eller förslag.

A.D.5 Uppgiftskategori
Ett område som en uppgift eller en uppgiftstyp kan relateras till.

A.D.6 Typ/kategori
I de fall där uppgiftstyp och/eller uppgiftskategori kan väljas genom en gemensam urvalslista används denna beteckning.

A.D.7 Merge
Begreppet används vid sammanslagning av uppgifter.

A.D.8 Dokumentation
Information i fritext eller fil som är kopplad till en uppgift.

A.D.9 Projekt
Ett fristående kundprojekt eller ett sammanhang att samla uppgifter under.

A.D.10 Resurs
Systemets användare. Representation av en person som är aktiv eller intressant i projekt. Interna och externa resurser är två grova nivåer. En egen anställd är en intern resurs medan kundrepresentanter, underkonsulter eller andra involverade personer är typiska exempel på externa resurser.

A.D.11 Föreslagen resurs
En uppgift kan fördelas till en resurs men det kan också vara ett förslag på flera resurser ifall det är oklart vem som ska ansvara för uppgiften.

A.D.12 Involverad resurs
En resurs som ingår aktivt i en projekt.

A.D.13 Användaren
En autenticerad resurs som använder systemet.

A.D.14 Kontakt
Vid hantering av e-post omnämns de personer användaren kan skicka svar till som kontakter. En kontakt är en representation av en person med en e-postadress och telefonnummer.

A.D.15 Användarföretag
Används för att gruppera resurser och projekt under det företag de tillhör.

A.D.16 Användargrupp
Används för att gruppera resurser med syftet att kunna tilldela dem olika rättigheter.

A.D.17 Adress
Motsvarar vanligtvis ett kontor för ett användarföretag.

A.D.18 Supportinbox
Definierar en koppling för att hämta in e-post från ett konto på en mailserver till Westimate.

A.D.19 Allmänt supportprojekt
I det projekt som har valts som allmänt supportprojekt för ett användarföretag samlas inkomna uppgifter från alla dess allmänna supportinboxar.

A.D.20 Matchning
En regel som definierar en katalog i ett projekt som e-post från vissa avsändaradresser ska placeras i.

A.D.21 Tidsrapport
En insats avseende ett antal timmars jobb av en resurs på en uppgift.

A.D.22 Aktivitet
Klassificering av vilken typ av arbete en tidsrapport avser.

A.D.23 Aktivitetskostnad
Timkostnad som anges för en aktivitet i ett projekt.

A.D.24 Event
En händelse i systemet, t ex när ett objekt skapas eller dess status förändras.

A.D.25 Notifikation
En uppmärksammande personlig referens till en resurs om en händelse.

A.D.26 UUID
En unik id-sträng.

A.D.27 EUID
En unik id-sträng i Microsoft Exchange.

A.1.4 Referenser

A.R.1 Scrum Development Process, Ken Schwaber, 1995.

A.R.2 Evidence Based Scheduling, Joel Spolsky, 2007. Från "Joel on Software", http://www.joelonsoftware.com/items/2007/10/26.html, hämtad i december 2007.

A.1.5 Översikt

Den andra delen av dokumentet beskriver allmänna krav, begränsningar på Westimate och vilka användarna är.

Den tredje delen av dokumentet ger en detaljerad förteckning över de mest strukturellt viktiga samt värdeskapande kraven som Westimate ska hantera. Slutligen redogörs det för miljön systemet ska verka i samt integration- och prestandakrav.

A.2 Allmänna krav

Den tekniska plattformen för systemets framtagande består av följande;

- HTML, CSS, Javascript, AJAX.
- Adobe Coldfusion. Applikationsserver med ett skriptspråk som serverkoden skrivs i.
- Farcry CMS. Ett framework och innehållshanteringssystem för Coldfusion.
- MySQL. Databas där systemets data lagras.

Westimate ska vara på svenska till att börja med, men dynamiskt språkstöd ska byggas in inför framtiden. Genom att lägga till en uppsättning strängar för ett annat språk och implementera en kontroll där de tillgängliga språken kan väljas i gränssnittet ska det vara löst. En applikation för att hantera dessa stränglistor och lagra dem i databasen finns redan på Westbahr från ett tidigare projekt.

För kostnadsangivelser på aktiviteter ska allting vara i SEK, värden utåt i andra valutor får räknas om manuellt eller med annat stöd utanför Westimate.

Westimates användare kommer främst vara interna resurser i form av anställd personal. Underkonsulter och kunder ska som externa resurser kunna följa arbetet i de projekt de fått tillgång till.

A.3. Specifika krav

A.3.1 Funktionella krav

A.3.1.1 Specifika krav relaterade till "Definitioner, förklaringar och förkortningar"

***** Uppgiftskategori
En kategori kan vara helt fristående eller kopplad till en uppgiftstyp.

***** "Merge"
Uppgifter som behandlar samma ärende ska kunna slås ihop. En uppgift ska anges som master och vara den som existerar i systemet, den eller de andra som slås ihop med mastern kommer att döljas och bara vara nåbara genom mastern. En mergad masteruppgift ska alltid markeras genom en stjärnsymbol efter titeln.

***** Föreslagen resurs
Ett förslag ska kunna göras tidbegränsat och en resurs som blir tilldelad uppgiften när förslaget förfaller ska kunna väljas ut.

***** Kontakt
En kontakt kan lagras en gång i en uppgift eller gemensamt för en kund. De som lagras

på en kund finns sedan tillgängliga från uppgifter i alla kundens projekt. Det samma gäller internt eftersom användarföretaget självt räknas som sin egen kund när kontakter läggs till på uppgifter i interna projekt.

* Användargrupp
Externa resurser behöver inte beröras av alla gruppens rättigheter.

***** Supportinbox
Ett användarföretag kan ha flera supportinboxar. En supportinbox kan vara allmän och det kan finnas flera allmäna inboxar. Mottagningen av dessa sker i ett gemensamt projekt; det allmänna supportprojektet. En supportinbox kan annars vara projektspecifik och då placeras de mottagna uppgifterna i ett särskilt projekt som väljs per inbox.

***** Matchning
För varje matchning finns det en regel, som är en lista av epostadresser eller ett reguljärt uttryck. Regeln styr vilka avsändaradresser som matchas. Inkommande allmänna supportuppgifter går igenom alla matchningar i en obestämd ordning och om avsändaren matchar så placeras uppgiften i den matchningens angivna projektkatalog. Om en avsändare skulle råka matcha flera regler är det alltså bara den som råkar testas först som väljs.

***** Aktivitetskostnad
Intern och extern kostnad varierar med aktivitet och det kan även finnas unika prisuppgifter per projekt eller kund.

***** Event / Notifikation
En notifikation är riktad till en resurs och den beskriver en händelse, så för ett event kan det skapas flera notifikationer.

***** Informationen ska presenteras i förklarande text. När vem gjorde vad och eventuellt vad som gällde innan denna förändring. Värdet före och efter ska sparas. Beroende på typen av händelse visas ett kortare eller mer utförligt meddelande om vad som skedde.

***** Det finns två olika medium för att notifiera. Notifikationsmail och en intern eventlog i Westimate. Händelser som en resurs utför ska aldrig resultera i ett mail till sig själv, men det ska alltid registreras i eventlogen. Om resursen är inkluderad i listan över de som ska notifieras i det projekt händelsen utfördes.

***** I en resurs inställningar ska det gå att ställa in vilka typer av händelser som ska gå ut via mail, oavsett om resursen är inkluderad i det aktuella projektets involverings- eller notifieringslista.

A.3.1.2 Gränssnitt

***** Gränssnittet ska bestå av en arbetsyta och en administrationsyta.

***** I arbetsytan behandlas projekt och uppgifter. Vyer ska finnas för att visa, filtrera, söka, ändra och skapa data i form av uppgifter i projekt.

***** I administrationsytan ska alla de inställningar kunna göras som påverkar vilken data och vilka alternativ som finns tillgängliga i arbetsytan, för de projekt som tillhör ett användarföretag. I det sällsynta fallet att en användare är intern resurs på fler än ett användarföretag, ska den användaren kunna ha tillgång till flera administrationsytor.

***** Systemet ska kunna hantera begränsande skillnader i gränssnitt för interna och externa resurser. Helst ganska fritt styrt genom rättigheter.

***** Per användargrupp ska det kunna anges om användarna har tillgång till dess

användarföretags adminstrationsyta.

A.3.1.2.1 Arbetsyta

Tanken är att 99% av tiden en användare spenderar i Westimate ska vara i arbetsytan.

Ytan ska var så fri som möjligt från inställningar som bara görs en gång och sedan används kontinuerligt. Många inställningar behöver inte visas eller vara redigerbara i det vanligare användandet av systemet utan placeras lämpligen i administrationsytan.

Ett exempel är valet över vilka resurser som ingår i ett projekt. Användaren kan ändå se vilka de är genom att se vilka resurser som uppgifter kan fördelas till, mer än så behöver inte visas eller vara redigerbart från arbetsytan.

Ett undantag är att användaren ska kunna skapa nya kontakter vid mailhantering. Samt att möjlighet ska finnas för användaren att ändra inställningar på sitt användarkonto, oavsett om de har tillgång till administrationsytan.

Nedan följer en genomgång av de delar som ska finnas i arbetsytan.

* **Översiktsvy**

- Användaren ska kunna skapa uppgifter direkt från översiktsvyn. Att välja projekt och eventuellt fördela till en av dess involverade resurser samt skriva en beskrivning ska räcka.

- Användaren ska kunna följa händelser i de projekt den har valt eller blivit tilldelad att bli notifierad om i en personlig eventlogg. Dessa inställningarna görs per projekt i administrationsvyn.

- Det ska finnas en månadskalender där utvalda datum har markerats. Utvalda datum ska vara de med tidsbestämda uppgifter på sig samt de där det finns uppgifter med ett uppskattat förfallodatum. Uppgifterna som hämtas in är från de projekt användaren är involverad i. Vid klick på ett datum visas länkar till de uppgifter som berörs av datumet.

- Det ska gå att få en sammanställning av de senaste uppgifterna användaren själv har utfört loggade händelser på. Det vore även intressant med listningar av ofärdiga uppgifter som 1) tillhör, 2) är föreslagen till och 3) har behandlats av användaren. Dessa olika uppgiftslistningar ska nog visas ihop i en vy eller en egen del av översikten, där andra urval kan bli aktuella i framtiden.

* **Projektvy**

- Katalog- och uppgiftstrukturen för projektet ska visualiseras, exempelvis genom ett träd. Uppgifter och kataloger ska kunna flyttas runt genom klick, drag och släpprörelser. Trädet ska minnas i vilken ordning objekten har placerats.

- Uppgifter och kataloger i en vald katalog ska listas i en tabell. Katalogerna ska visas överst, användaren ska kunna ändra namn på dem genom någon symbol och framförallt kunna klicka på deras titel för att öppna deras innehåll i tabellen. Under katalogerna kommer den valda katalogens uppgifter. Deras titel ska kunna klickas för att öppna en annan vy med detaljerna om uppgiften.

- Den valda katalogen ska markeras och öppnas i trädet vid sidomladdning. Det ska även finnas en så kallad "breadcrumb" navigation längst upp på sidan där det redogörs för vilken katalog som är i fokus, och vilken väg igenom katalogerna i hierarkin som leder dit.

- Varje rad med en uppgift i tabellen i projektvyn ska kunna ha kolumner för att visa senaste ändring, antal inkomna/utgående mail, antal kommentarer, typ/kategori, tilldelad resurs och status. De tre sistnämnda ska kunna ändras genom att byta värde direkt från tabellvyn. Sedan ska det kunna finnas kolumner för att ändra uppgiftens tidsuppskattning och för att lägga till en tidsrapport. Vilka kolumner som visas för tillfället kommer nog konfigureras allt eftersom, alla kanske inte kommer användas.

- Framför uppgiftens titel i tabellen ska en färgsymbol användas för att visa prioritet. Låg är gul, normal är orange och hög är röd.

- Över tabellen ska det finnas möjlighet att filtrera tabellen efter kriterier som klara/ oklara uppgifter och typ/kategori. Denna filtermodul ska även kunna påverka trädet. Sortering av uppgifterna i tabellen kan också placeras här. Inställningarna här ska sparas i sessionen så att de inte tappas om användaren går ifrån och återvänder till projektvyn.

- I det allmänna supportprojektet för användarens användarföretag ska uppgifter som varit avklarade i en längre tid och legat kvar visas i en tabell under den ordinarie där dessa uppgifterna uppmanas att arkiveras till en kunds defaultprojekt. Detta val ska kunna göras per uppgift, sedan godkänns alla valen genom en knapp.

- Det ska även finnas möjlighet att snabbt lägga in en uppgift på liknande vis som på översiktsvyn, med förändringen att projektet ju behöver inte väljas. Bara resurs och beskrivning. Dessutom ska det finnas en tydlig knapp för att kunna skapa en ny uppgift i detalj via uppgiftsvyn.

- Allt detta ovan beskriver projektvyns normalläge, visningen av uppgifter. Vid sidan om denna ska det finnas en undermeny för att istället för listningen av uppgifterna kunna visa en annan aspekt av projektet. T ex projektets statistik, filer från projektets uppgifter eller en gemensam eventlogg för projektet.

* **Projektstatistik**

Vad som presenteras här kommer garanterat förändras när systemet är i användning. De tankar på användbar statistik som finns för tillfället är:

- Förteckning över antalet uppgifter inom varje typ/kategori och status över en valfri tidsperiod (med föregående vecka som grundinställning).

- Förteckning över hur länge uppgifter har lämnats oavklarade utan att avslutas. Visa detta i antal uppdelat i intervall på till exempel några dagar, någon vecka och längre än en månad.

- Ett processdiagram (eng: Burndown chart) [A.R.1] som följer det totala antalet uppgifter i projektet samt andelen som är klara eller inte.

- Evidence Based [A.R.2] Probability Chart med sannolikhetsfördelning över antalet timmar som projektets estimerade uppgifter bör kräva innan de är klara.

- Evidence Based [A.R.2] Probability Chart History - historik över hur procentvärden för sannolikhetsfördelningen har förändrats.

* **Uppgiftsvy**

- En uppgifts innehåll ska vara tillgängligt att redigera och arbeta med i en komprimerad vy med olika moduler eller uppdelat på olika sidor med hjälp av flikar. Den komprimerade vyn ska i minsta fall innehålla en sammanställning av det viktigaste - som redogörs för nedan, vad som ska täckas in får undersökas vidare när systemet är i

användning. Tanken är att den komprimerade vyn kommer vara att föredra av avancerade användare som känner systemet medan nya användare kommer kunna använda flikarna för att lättare hitta funktionerna och få dem presenterat tydligare var för sig.

- Det redigerbara innehållet ska minst bestå av rubrik, beskrivning, tilldelning eller förslag på ansvarig resurs, status, prioritet, typ/kategori, start/slut datum och om de ska visas i kalendern, tidsuppskattning i timmar och/eller beräknat slutdatum, tidsrapporter, mail, kommentarer och filer.

- Det ska gå att flytta uppgiften till ett annat projekt.

- Det ska gå att sammanfoga uppgiften till en annan uppgift.

- En eventlogg ska finnas, som visar händelser i beskrivande text. Vem som utförde vad. Vid enklare ändringar vore det bra om det skrivs ut vad värdet var innan det ändrades också. Större datamängder som beskrivningen kanske ska kunna nås genom en dold ruta, popup eller dylikt. Vid ändringar ska moduler eller enskilda kontroller vara separerade och registrera ändringarna individuellt så att användarna kan följa i detalj när och i vilken ordning allt skedde. Om en modul är separerad ska det markeras när den har osparade ändringar. Om det är en enskild kontroll som är separerad ska ett meddelande visas automatiskt när ändringen har gjorts och sparats.

- Uppgiftens datum för när den senast blev ändrad ska täckas in av allt ovanstående undantaget när den flyttas till ett annat projekt.

- Mailmodulen ska klara av att skicka ett mail till flera kontakter. Det ska finnas stöd för en signaturmall per supportinbox som ersätts per användare, och det ska gå att hämta in historik från ett tidigare mail i konversationen. Hela mailhistoriken ska vara tillgänglig. Bifogade filer ska visas i inkommande mail och det ska gå att bifoga filer till mail.

- Filmodulen kan användas för att ladda upp filer. För att maila nya filer själv behöver användaren ladda upp dem här först. Filer som bifogats i inkommande mail ska också visas här.

- Om uppgiften inte kan hittas ska det visas en vy med ett felmeddelande. Om det finns eventhistorik för en uppgift med detta UUID ska den visas. Historiken ska finnas kvar ifall uppgiften har tagits bort. Framtida funktionalitet här skulle kunna vara att återställa en borttagen uppgift.

* **Sökning**

- Det ska gå att söka efter uppgifter baserat på rubrik, beskrivning, kommentarer och mailinnehåll. Börja med en enkel sökfunktion. Längre fram kommer det behövas en bättre lösning, exempelvis genom att använda Verity eller Lucene. Även indexering av några vanliga filtypers innehåll kan vara intressant på sikt.

* **Inställningar**

- Användarna ska kunna ändra inställningar för sitt konto. Ange kontaktinformation, som telefonnummer och epostadress, samt kunna ändra lösenord. De ska kunna ange vilka typer av händelser de vill bli notifierade om via mail De ska kunna ange uppgifterna för sitt Exchangekonto.

A.3.1.2.2 Administrationsyta

De ändringar som görs i administrationsytan görs ofta väldigt snabbt och sällan. De objekt som hanteras här ligger till grund för begränsningar och vilka data som finns

tillgängligt i det motsvarande användarföretagets arbetsyta.

Genom enkla listor och avgränsade editeringsvyer bör det mesta i administrationsytan kunna hanteras. Här följer en kortfattad redogörelse för vad:

***** kunna ändra grundläggande data för användarföretaget (namn och organisationsnummer)

***** kunna hantera supportinställningar
- kunna välja allmänt supportprojekt samt vilka resurser som ska underrättas med mail när det kommer in allmänna supportuppgifter
- kunna skapa/ändra/radera supportinboxar
- kunna skapa/ändra/radera matchningar

***** kunna skapa/ändra/radera adresser

***** kunna skapa/ändra/radera interna kontakter

***** kunna hantera kunder
- kunna skapa/ändra/radera kunder (namn, kundnummer och organisationsnummer)
- per kund: kunna skapa/ändra/radera adresser
- per kund: kunna skapa/ändra/radera kontakter

***** kunna hantera projekt
- kunna skapa/ändra/radera projekt (per kund och även interna projekt)
- per projekt: kunna välja involverade resurser (de som kan tilldelas uppgifter)
- per projekt: kunna välja vilka resurser som notifieras om händelser i projektet i den personliga eventloggen i översiktsvyn
- per projekt: kunna ange aktivitetskostnader, alla aktiviteter visas
- per projekt: kunna koppla externa resurser (med UUID, e-post eller personnummer)

***** kunna skapa/ändra/radera användargrupper

***** kunna skapa/ändra/radera interna resurser

***** kunna skapa/ändra/radera aktiviteter

***** kunna skapa/ändra/radera aktivitetskostnader

***** kunna skapa/ändra/radera uppgiftstyper

***** kunna skapa/ändra/radera uppgiftskategorier

A.3.1.2.3 Övrigt gränssnitt

*** Navigationsmeny**

Längst upp i gränssnittet ska man kunna välja mellan att se översikten eller projekvyn för ett projekt. Till projektväljaren behövs ett kundfilter för att enklare hitta bland projekten. Sedan ska det finnas länkar till andra vyer och funktioner såsom sökning, användarkontoinställningar, administrationsvy och utloggning.

*** Inloggning / autentisering**

Det ska finnas funktionalitet för att logga in och ut ur systemet. Försöker en resurs nå en vy utan att vara autensierad ska inloggningsvyn visas, för att sedan slussa vidare användaren till den önskade sidan om inloggningen godkänns. Notera att eventuella skript eller vyer i systemet som ska vara nåbara utan autentisering måste kunna

exkluderas från detta.

A.3.2 Integreringskrav

A.IK.1 Sändning och mottagning av supportmail

En eller flera mailboxar behöver skapas per användarföretag. De ska stödja Post Office Protocol version 3 (POP3) eftersom <cfpop> i Coldfusion bör vara den metod som används för att ta emot mail. Även <cfmail> som används för att sända mail kräver i nuläget att kommunikationen med mailservern går över POP3.

A.IK.2 Microsoft Exchange

Information och dokumentation om projekt och kunder ska vara enkelt åtkomlig för tekniker med mobil tillgång till Exchange. Genom att använda tvåvägskommunikation med hjälp av Microsoft Exchange-objekt kan en transparens mellan data i Westimate och de vanligaste fälten som är tillgängliga genom Exchange uppnås.

Uppgifter ska kunna exporteras som Exchange Tasks. Data som (Subject, Message, Priority, Status, PercentCompleted, ActualWork, TotalWork, DueDate) exporteras och läses in. (Attachments) också om det är enkelt ordnat. Övriga data finns kvar i Westimate som är master.

En senare uppdateringsexport till samma Exchange Task ska bara beröra de tidigare nämnda attributen. Westimate ska automatiskt försöka genomföra denna uppdatering när en uppgift ändras, men inga mer synkningsåtgärder än så behöver vidtas. Westimate ska ej kunna importera eller synka uppgifter som skapas eller sparas genom Microsoft Exchange. Det ska vara envägskommunikation för att förenkla detta.

A.IK.3 Applikationer på intranätet

Westimate ska placeras i företagets intranät. På intranätet ligger i nuläget en wiki, ett enkelt informationssystem och ett tidsrapporteringssystem.

Westimate ska verka brevid detta system. Tidsrapporering som sker på uppgifter i Westimate ska vara tillgängligt för tidsrapporteringssystemet. Detta blir dock inte mer avancerat än att läsa från databasen och omforma datan till rätt format, men detta ska ske i tidsrapporteringssystemet.

Det fanns även ett enkelt informationssystem för hantering av miljöer, servrar, tjänster, konton med mera som knappt användes men som det kommer bli ett allt större behov för. Antingen behövde detta byggas på och integreras eller helt flyttas in i Westimate på sikt. Läget är oklart.

A.3.3 Prestandakrav

A.PK.1 Mottagning av supportmail

Inkommande mail behöver hämtas in och behandlas snabbare än takten för hur nya mail inkommer. Behandlingen av ett mail bör i genomsnitt ej ta mer än någon sekund. Mail med stora bifogade filer hör till ovanligheterna. Nedhämtningen av mail kommer att schemaläggas och till och med med ett kort intervall på en minut så bör 15-20 mail i minuten inte vara några problem att hantera med 3-4 sekunder per mail. Genom att sänka intervallet kan toleransen höjas. Om en nedhämtning initieras innan den föregående har avslutats så ska den nya ej påbörjas. Till nästa försök bör mailkön ha

reducerats så att den omgången går igenom smidigt.

Antalet mottagande inboxar bör inte bli ett problem förrens det handlar om många tiotal. Tiden det tar att behandla ett mail borde vara större än tiden för att ansluta till en mailbox och se om där finns några nya mail.

A.PK.2 Kommunikation med Exchange-server

Behandlas utav Coldfusion och ska endast ske för ett objekt åt gången vilket ej bör leda till några fördröjningar som behöver täckas upp.

A.PK.3 Responstider i gränssnittet

Sidladdningar på över en sekund ska höra till undantagen, målet är att genomsnittet bör hamna omkring en halv sekund.

Till undantagen hör vyer såsom översikten, projektvyn och sammanställningen över uppgifter. Eventuellt behöver mycket data hämtas och behandlas åt gången. Databaslogik och nästlade loopar får givetvis ej göras exponentiella. Laddningstiden ska vara så linjär som möjligt. Prioritera dock inte optimeringar framför att undvika redundans.

För asynkrona Javascript-anrop bör genomsnittet vara under en halv sekund. Det ska inte vara tyngre processer inblandade utan som mest ett par databasfrågor och enkel behandling av dem. Dock kan sändning av mail och utbyte med Exchange behöva vänta på att processen lyckats innan gränssnittet uppdateras, i andra fall av mindre betydelse kan det trådas ut.

A.4. Förvaltningskrav

Från visionen med Westimate framgår det verksamhetsnära användandet av systemet. Det måste antas att Westimate kommer att utvecklas och förändras så länge det används.

Denna del av specifikationen har knappt hunnits med, både anledningen bakom och konsekvensen av detta är att vidare utveckling av systemet är svår att förutsäga. Enhetstestning bör användas så mycket som möjligt, i övrigt med undantag för några reservationer inför framtiden tvingas denna del lämnas ofullständig tills vidare.

A.4.1. Specifika förvaltningsbarhetskrav

Några uppenbara reservationer för framtidsplaner har identifierats:

A.SFK.1 Produktvision

Westimate bör ha potential att bli en tjänst som kan säljas kommersiellt. Med potential menas att det inses att det kommer krävas en ytterligare insats efter detta inledande arbete för att ta systemet till denna kommersiella nivå. Främst inom användarvänlighet, stabilitet och säkerhet, något som det inte finns möjlighet att garantera kvalitén på med denna första insatsen. Med implementationen som täcks av detta arbete kan personer på företag som står Westbahr nära och som är involverade i något enstaka projekt använda Westimate. som externa resurser med begränsad åtkomst. Det som menas med kommersiell tjänst här är i princip att upplåta en egen instans av Westimate som kunden själv rår över - inte bara den externa resurskopplingen.

A.SFK.2 Projektekonomi

Ett annat mål är att implementera projektekonomi längre fram. Genom att räkna på intäkter och åtgången tid för olika aktiviteter för att få ett mått på lönsamhet.

A.SFK.3 Regioner

På sikt bör projekt och kunder kunna fördelas bland regioner. En adress för ett kontor kan tänkas definiera en region. Adresser kommer behövas när funktionalitet för projektekonomi byggs till framöver.

A.SFK.4 Event

Kommer antagligen endast beröra händelser på uppgifter även i framtiden, men ska redan nu implementeras generellt så att det går att utöka med händelser på andra objekt ifall det behovet uppstår.

A.SFK.5 Versioner

Hantering av olika "releases" och versioner i projekten kan komma att behövas på sikt.

A.SFK.6 Parallell användning

Det behöver införas ett skydd mot när flera användare behandlar samma uppgift samtidigt.

Appendix B: Källkod

I denna bilaga presenteras källkoden för algoritmens huvudsakliga metoder. Några mindre modifikationer har gjorts för att korta ner raderna och höja läsbarheten:

- Anrop till metoder i utomstående klasser enligt *application.oClass.method()* har förkortats till bara *method()*.
- Ett lokalt *scope* som har använts för lokala variabler i metoderna, enligt *local.variable*, har uteslutits ur koden - bara *variable* skrivs ut.

B.1 generateMonteCarlo()

```
<cffunction name="generateMonteCarlo" returntype="array" output="false" access="public">
   <cfargument name="projectid" type="uuid" required="true">
   <cfargument name="ROUNDS" type="numeric" required="false" default="100">
   <cfscript>

   // Get factor arrays for INTERNAL resources involved in the project!
   stAE = {};
   qResources = getInvolvedResourcesByProject(projectid=session.platform.project,includeExternal=0);
   for (i=1; i<=qResources.RecordCount; i++) {
      rid = qResources.objectid[i];
      StructInsert(stAE, rid, getFactorsArrayByResource(rid=rid).array);
   }

   // Get available estimates etc for tasks in the project (or folder)
   aTasks = getTasksDataArray(projectid=arguments.projectid);

   // Do this for a number of times, each associated with a 1/n % probability
   aMonteCarlo = [];
   for (i=1; i<=arguments.ROUNDS; i++) {

      // Get a value in projected time for this run
      aMonteCarlo[i] = 0;
      for (j=1; j<=ArrayLen(aTasks); j++) {

         if (aTasks[j].estimated GT 0 AND NOT aTasks[j].completed) {

            rid = aTasks[j].ownerRID;
            if (isValid("uuid",rid)) { // assigned tasks
               /* handle the case where a task is assigned to a resource
               // that is not included in the projects involved list atm */
               if (NOT StructKeyExists(stAE, rid)) {
                  StructInsert(stAE, rid, getFactorsArrayByResource(rid=rid).array);
               }
            }
            else { // distribute unassigned tasks by how the assigned tasks are distributed proportionally
               // count num of tasks total and per resource, for the distribution below
               if (NOT StructKeyExists(local,"distribution")) {// is only run the first time this else is reached
                  distribution = setupOwnerDistribution();
               }
               ridpick = getRandomOwnerByDistribution();
```

```
            if (ridpick EQ -1) { // pick a resource randomly
                rid = ListGetAt( StructKeyList(stAE), RandRange(1,ListLen(StructKeyList(stAE))) );
            } else {// somewhat randomly pick weighted by the assignment distribution of the project
                rid = ridpick;
            }
        }

        // randomly select one of the resources factors
        factor = stAE[rid][ RandRange(1, ArrayLen(stAE[rid])) ];

        // adjust estimate
        adjustedEstimate = factor * aTasks[j].estimated;

        // handle already reported time
        if (aTasks[j].reported GT 0) {
            if (aTasks[j].reported GTE adjustedEstimate) {
                adjustedEstimate = 0;
            }
            else {
                adjustedEstimate = factor * abs(aTasks[j].estimated - aTasks[j].reported);
            }
        }

        // add the calculated value to this runs total estimation
        aMonteCarlo[i] += adjustedEstimate;
    }
  }
}
</cfscript>
<cfreturn aMonteCarlo>
</cffunction>
```

B.2 plotMonteCarlo()

```
<cffunction name="plotMonteCarlo" returntype="void" access="public" output="true">
    <cfargument name="aMonteCarlo" type="array" required="true">
    <cfargument name="plotMaxValue" type="numeric" required="false" default="95">

    <cfset var i = 0>

    <cfset ArraySort(arguments.aMonteCarlo,'numeric')>
    <cfset min = arguments.aMonteCarlo[1]>
    <cfset max = arguments.aMonteCarlo[ArrayLen(arguments.aMonteCarlo)]>

    <cfif (max - min EQ 0) OR (min EQ 0)>
        <!--- bad values, show empty plot --->
        <cfoutput>
            <cfchart style="silver" gridlines="0" chartwidth="600">
            </cfchart>
        </cfoutput>
        <cfreturn local>
    </cfif>

    <cfoutput>
        <cfchart style="silver" gridlines="0" chartwidth="600">
```

```
        <cfchartseries type="line" markerStyle="circle">

            <!---
            Plot zeroes from 0h to the lowest hour value (that represents 5%).
            Increase the step value accordingly to the distance between max and min
            --->
            <cfset step = 1 + round((max-min)/100)>
            <cfloop index="i" from="0" to="#min#" step="#step#">
               <cfchartdata item="#i#h" value="0">
            </cfloop>

            <!---
            Plot results scaled
            --->
            <cfset scaled = ArrayNew(1)>
            <cfset procentlo = 5.0>
            <cfset procenthi = 95.0>
            <cfloop index="i" from="1" to="#ArrayLen(arguments.aMonteCarlo)#">

               <cfset scaled[i] = arguments.aMonteCarlo[i]>
               <cfset scaled[i] -= min>
               <cfset scaled[i] *= (procenthi-procentlo)/(max-min)>
               <cfset scaled[i] += procentlo>
               <cfset scaled[i] = NumberFormat(scaled[i], "__.")>
               <cfchartdata item="#arguments.aMonteCarlo[i]#h" value="#scaled[i]#">

            </cfloop>

         </cfchartseries>
      </cfchart>
   </cfoutput>
</cffunction>
```

B.3 setupOwnerDistribution()

```
<cffunction name="setupOwnerDistribution" returntype="struct" access="public" output="true">
   <cfscript>
   distribution = {};
   distribution.total = 0; // num of tasks
   distribution.totalowned = 0; // num of assigned tasks
   distribution.owned = {}; // num of assigned tasks per resource
   for (k=1; k<=ArrayLen(aTasks); k++) {
      distribution.total += 1;
      if (NOT StructKeyExists(distribution.owned,aTasks[k].ownerRID)) {
         distribution.owned[aTasks[k].ownerRID] = 0;
      }
      if (isValid("uuid",aTasks[k].ownerRID)) {
         distribution.totalowned += 1;
         distribution.owned[aTasks[k].ownerRID] += 1;
      }
   }
   </cfscript>
   <cfreturn distribution>
</cffunction>
```

B.4 getRandomOwnerByDistribution()

```
<cffunction name="getRandomOwnerByDistribution" returntype="string" access="public" output="true">
   <cfargument name="bOnlyOwned" type="numeric" required="false" default="0">
   <cfscript>
   if (arguments.bOnlyOwned) {
      // do not care about how many of the tasks that aren't owned by someone
      // calculate the distribution among the owned tasks. (so if there is only
      // one owned task and many unassigned - all the unassigned will with 100%
      // probability be used in the model with that owners stats)
      randomvalue = RandRange(1, distribution.totalowned);
   }
   else {
      // include the tasks without an owner in the distribution and randomize
      // that case among all the resources with a probability based on how many
      // ownerless tasks there are
      randomvalue = RandRange(1, distribution.total);
   }

   accusum = 0;
   ridpick = -1;
   for (ridindex in distribution.owned) {
      accusum += distribution.owned[ridindex];
      if (randomvalue LTE accusum) {
         ridpick = ridindex;
         break;
      }
   }
   </cfscript>
   <cfreturn ridpick>
</cffunction>
```

B.5 getFactorsArrayByResource()

```
<cffunction name="getFactorsArrayByResource" returntype="array" access="public" output="false">
   <cfargument name="rid" type="uuid" required="true">
   <cfargument name="minFactors" type="numeric" default="#ArrayLen(getRandomFactors())#">
   <cfargument name="maxFactors" type="numeric" default="100">

   <cfset array = ArrayNew(1)>
   <cfset query = getTaskFactorsQueryByResource(
      rid=arguments.rid,
      maxFactors=arguments.maxFactors)>

   <cfloop query="query">
      <cfif isNumeric(reported) AND reported GT 0 AND isNumeric(estimated) AND estimated GT 0 >
         <cfset ArrayAppend(array, reported/estimated)>
      </cfif>
   </cfloop>

   <!--- If very few factors are retrieved above, add some random factors to compensate --->
   <cfloop array="#getRandomFactors()#" index="randomfactor">
      <cfif ArrayLen(array) LT arguments.minFactors>
         <cfset ArrayAppend(array, randomfactor)>
      <cfelse>
```

```
            <cfbreak>
        </cfif>
    </cfloop>

    <cfreturn array>
</cffunction>
```

B.6 getTaskFactorsQueryByResource()

```
<cffunction name="getTaskFactorsQueryByResource" returntype="query" access="public" output="false">
    <cfargument name="rid" type="uuid" required="true">
    <cfargument name="maxFactors" type="numeric">
    <cfset var q = "">
    <cfquery name="q" datasource="#application.dsn#">
    SELECT t.objectid , t.subject, (SELECT targethours FROM wbTaskEstimation WHERE taskid = t.objectid
ORDER BY datetimecreated DESC LIMIT 0,1) AS estimated, (SELECT SUM(hours+hoursextra) FROM
wbTimeReport WHERE taskid = t.objectid GROUP BY taskid) AS reported
    FROM wbTask AS t
    WHERE ownerRID = <cfqueryparam cfsqltype="cf_sql_varchar" value="#arguments.rid#">
    AND status = "Completed"
    HAVING (estimated IS NOT NULL AND estimated<>'' AND reported IS NOT NULL AND reported<>'')
    ORDER BY t.datetimelastupdated DESC
    LIMIT 0,#arguments.maxFactors#
    </cfquery>
    <cfreturn q>
</cffunction>
```

B.7 getRandomFactors()

```
<cffunction name="getRandomFactors" returntype="array" output="false" access="public">
    <cfset var a = ArrayNew(1)>
    <cfset a = [ 0.3, 0.5, 0.8, 1.0, 1.1, 1.2, 1.3, 1.4, 1.6, 1.8, 2.0, 2.2, 2.5, 3.0, 4.5]>
    <cfset CreateObject("java","java.util.Collections").Shuffle(a)>
    <cfreturn a>
</cffunction>
```