Högskolan i Halmstad
IT-projekt 10p.
VT 07

Hälsokoll

En virtuell coach

Handledare: Mats Lindqvist

Författare:
Jacob Wodzynski, 851230
Matija Prskalo, 860714
Frida Jantell, 820921

1 Inledning

Med ökande påtryckningar från samhället, om att människor hela tiden ska prestera i allt högre tempo, kan många idag uppleva stress. Som resultat av detta kan vara att somliga hoppar över måltider eller köper snabbmat istället för att laga själva. Dessutom struntar många i motion, i syfte att hinna utföra mer saker. Media framställer dagligen nya, ”fantastiska” metoder för att minska vikten, och människors hälsa har kommit att bli mer och mer uppmärksammad. Många personer lider av övervikt, samtidigt som alltfler är undernärda och får ingen kontroll på sitt oregelbundna kostintag.

Hälsokoll är en tjänst som är framtagen för att hjälpa människan i sitt dagliga liv för att uppnå en bättre hälsa. Tjänsten erbjuder bl.a. kostråd, motionstips och recept; allt det speciellt utformat efter användarens profil. Med vår tjänst önskar vi att människor ska kunna må bättre, bli piggare och förhoppningsvis, som resultat av detta, leva längre.

1.1 Syfte

Syftet med projektet är att utforma en fungerande prototyp av en tjänst som underlättar användarens liv genom att hålla koll på dennes hälsa. Hälsokoll är tänkt att vara en tjänst som finns både på webben och i mobilen.

Det är inte tänkt att, under projektets gång, bygga upp en prototyp som inkluderar alla de delar som är tänka att ingå i den slutliga versionen av tjänsten. Många områden kommer endast att beskrivas teoretiskt och själva implementeringen kan komma att ske vid ett senare tillfälle.

1.2 Metod

För att kunna genomföra projektet har vi som åtanke att söka stöd i litteratur från diverse områden, så som projekthantering, systemutveckling, marknadsföring, databashantering och programmering. Den lämpliga litteraturen kommer att hittas dels genom egen sökning i bibliotek och artikeldatabaser, och dels genom samtal med experter och handledare.

Inledningsvis kommer litteratur som berör projekthantering och systemutveckling att studeras, i syfte att hitta ett bra arbetssätt samt finna stöd till utvecklingsprocessen av den tilltänkta prototypen. Därefter kommer kunskap från andra områden, som databashantering och webbutveckling att inhämtas, för att kunna bygga själva tjänsten.

Gruppen kommer under test av prototypen att använda sig av kvalitativa metoder som, enligt Backman (1998), kännetecknas av att de inte använder sig av siffror eller tal utan istället resulterar i verbala formuleringar.

2 Teori

2.1 Projektarbete

Ett projekt defineras av Pearlson och Saunders (2006) som en tillfällig ansträngning åtagen för att skapa en unik produkt. Tillfälligt innebär att varje projekt har en fastställd början och ett fastställt slut. Unik innebär att produkten skiljer sig på ett utmärkande sätt från liknande produkter.

Enligt Eklund (2002) växer en projektgrupp samman först då ett samspel mellan individerna frodas. Författaren definierar, precis som Andersen, Grude och Haug (1994), att ett projekt bl.a. kan beskrivas som engångskaraktäristiskt, målinriktat, avgränsat i tid och omfattning, tilldelat begränsade resurser och uppdelat i delmål. En projektgrupp består av en projektledare, som enligt Carlson och Nilsson (2002) ansvarar för att leda projektgruppen, styra projektet mot det tänkta målet och se till att arbetet blir dokumenterat. Övriga projektmedarbetare utgör en viktig resurs genom att bidra med sin kompetens.

Svårigheter med att arbeta i projekt, som Eklund (2002) nämner, är att projektgruppens medlemmar besitter olika bakgrunder, erfarenheter, kunskaper och värderingar vilket kan leda till konflikter. Däremot bidrar individernas olika egenskaper till dynamik och goda resultat då de olika kunskaperna kompletterar varandra.

Enligt Eklund (2002) kan förlorandet av kontroll vara det värsta som drabbar ett projekt. Finns inte kontroll erhålls därmed inte möjligheten att styra projektet förbi svårigheter. För att undvika förlorad kontroll gäller det att ha en underliggande modell eller plan att arbeta efter, och med denna modellen kan sedan uppföljning av projektet göras.

Under ett projekt hålls det många olika möten både med projektgruppen och uppdragsgivaren beskriver Eklund (2002). Dessa möten ses som viktiga källor som ger upphov till diskussioner där alla involverade har chansen att tala fritt. Möten hålls även för att lösa problem och fatta beslut. Dessutom kan regelbundna sammanträden bidra till att motverka konflikter (McManus & Wood-Harper, 2003).

Det finns många skäl till varför projekt inte blir klara i tid. Där dålig planering är den vanligaste orsaken enligt både Eklund (2002) och Andersen et al. (1994). Andersen Författarna menar att problemet vid planeringen är att denna antingen blir för grov eller för detaljerad. I verkligheten behövs det minst två planeringsnivåer, dels en översiktsplan som beskriver vad som ska uträttas, vilken används då projektet diskuteras med beställaren. Dessutom behövs en lite mer detaljerad plan som visar hur olika delar av projektarbetet ska utföras. Denna syftar till att stödja projektmedlemmarna under arbetets gång. Eklund (2002) menar att projektplanen skrivs för projektgruppens skull, för att inom gruppen klargöra ansvarsområden, konkretisera tidsplaner och beskriva projektets mål, etc. Tidsplanen kan tas fram i form av ett Ganttschema (ibid.). Ganttschemat är ett populärt sätt att illustrera startdatum och slutdatum för de respektive uppgifterna som ska utföras i projektet [5].

Utifrån planeringen bör kontroller göras i form av uppföljning, där den aktuella situationen beskrivs enligt Eklund (2002). Uppföljning är till för att analysera situationen och möjliggöra korrigering i planeringen enligt Andersen et al. (1994).

2.2 Riskanalys

"Riskidentifiering innebär att hitta de faktorer som kan påverka tidsplanen i negativ riktning" (Marttala & Karlsson, 1999. s. 114). Till att börja med skrivs riskerna ned i form av en lista (ibid.). Nästa steg innebär att en riskanalys ska göras i vilken en bedömning hur varje identifierad risk kan påverka tidsplanen. Enligt Avison och Fitzgerald (2003) består en riskanalys av följande punkter:

- Identifiering av områden som kan utsättas för potentiella risker
- Bedömning och lokalisering av riskmöjligheter
- Identifiering av respons till risker
- Lokalisering av kostnader för risk och riskhantering

Resultatet av analysen blir en kompromiss mellan förväntad risk och förväntade kostnader som ger underlag för utformningen av en riskhanteringsstrategi som tillhandahåller olika tillvägagångssätt riktade mot att ta hand om specifika riskkällor (Avison & Fitzgerald, 2003).

Enligt Sommerville (2004) kan förekomsten av risker vara ett hot mot projektet, mjukvaruutvecklingen och organisationen. Författaren menar att det finns tre relaterade kategorier av risk:

1. Projektrisk – påverkar projektschemat eller resurserna.
2. Produktrisk – berör kvalitén eller prestandan hos mjukvaran.
3. Företagsrisk – har verkan på organisationen som utvecklar eller uppbringar mjukvaran.

Andersen et al. (1994) påpekar några förhållanden som måste beaktas vid bedömning av risker:

- Själva planen kan medföra risker
- De yrkesmässiga riskområdena måste identifieras
- Besluten är ofta kritiska för framtiden
- Beräkningarna av resursbehovet är alltid ett riskområde
- Resurstillgången är ofta osäker i ett projekt

2.3 Utvecklingsmetoder

Termen *life cycle model*, även kallad livscykelsmodell eller utvecklingsmodell, används för att representera ett antal aktiviteter och hur de är relaterade till varandra (Sharp, Preece & Rogers, 2007). Enligt Sommerville (2004) är en utvecklingsmodell

en förenklad beskrivning av en mjukvaruprocess som presenterar ett synsätt på denna. (ibid.).

De mer sofistikerade utvecklingsmetoder innehåller även en beskrivning av hur och när nästa steg ska tas (d.v.s. hur och när det är dags att förflytta sig till nästa aktivitet i modellen) samt vad som ska levereras ("komma ut") från varje steg (Sharp et al., 2007). Författarna nämner vidare att anledningen till att utvecklingsmetoder är populära är följande: a) de ger utvecklarna och - framför allt - beslutsfattarna möjligheten till att se en överblick över utvecklingsarbetet, vilket i sin tur ger dem möjlighet till att, b) följa händelseförloppet i utvecklingen, specificera vad varje steg ska leverera, allokera resurser, sätta mål, med flera.

Sharp et al. (2007) menar på att de utvecklingsmetoder som används idag varierar stort i sofistikeringsgraden och komplexiteten. För projekt med få, erfarna utvecklare, skulle antagligen en enkel process vara passande. För stora projekt däremot, med flera hundra utvecklare och kanske flera tusen användare, krävs något mer än en enkel process för att leverera ett välutvecklat system. Det krävs något som ger utvecklarna stöd, och tillhandahåller mer formalitet och disciplin. Sharp et al. (2007) säger att oavsett om en utvecklingsmetod är enkel eller komplex, är den i slutändan en förenklad version av verkligheten. Den är menad som en abstraktion, och alla välgjorda abstrakta representationer av verkligheten bör endast innehålla den mängden detaljer som krävs för den specifika uppgiften som de ska hjälpa till att lösa. Vidare menar författarna på att alla som önskar använda sig av en utvecklingsmetod kommer att behöva lägga till specifika detaljer för att passa de rådande omständigheterna.

Mjukvaruutveckling har under årens lopp resulterat i ett stort antal utvecklingsmetoder. Några exempel är de traditionella *vattenfallsmodellen* och *spiralmodellen*, samt de nyare, lättrörliga – från *agile* (snabb, rörlig, vig) – utvecklingsmetoder, som DSDM (*Dynamic Systems Development Method*) och XP (*eXtreme Programming*). De utvecklingsmetoder som kommer att presenteras i avsnitten nedan är enligt Sharp et al. (2007) värda att nämnas, då de:

- representerar de modeller som används ute i näringslivet och har visat sig vara framgångsrika
- visar hur tungvikten inom mjukvaruutveckling under senare år gradvis har flyttats åt interaktionshållet och en användarcentrerad synpunkt på utveckling

Påståendet i andra punkten delas även av Sommerville (2004), då författaren skiljer utvecklingsmetoderna i *The waterfall approach* och *Evolutionary Development*. Dessutom ses ännu en grupp, som kallas för *Component-based software engineering*. För att sammanfatta detta, anser Sommerville (2004) att de flesta utvecklingsmetoder är baserade på någon av de tre tankesätten:

1. *The waterfall approach* – Aktiviteterna som ingår i livscykeln representeras som separata processer, som t.ex. kravspecifikation, mjukvarudesign, implementering, tester, etc. Efter att varje steg är avklarat kan det "bockas av" och utvecklingen fortsätter till nästa.
2. *Evolutionary Development* – I den här processen sker ett ständigt "hoppande" mellan aktiviteter, såsom kravframtagning, utveckling och testning/validering (för

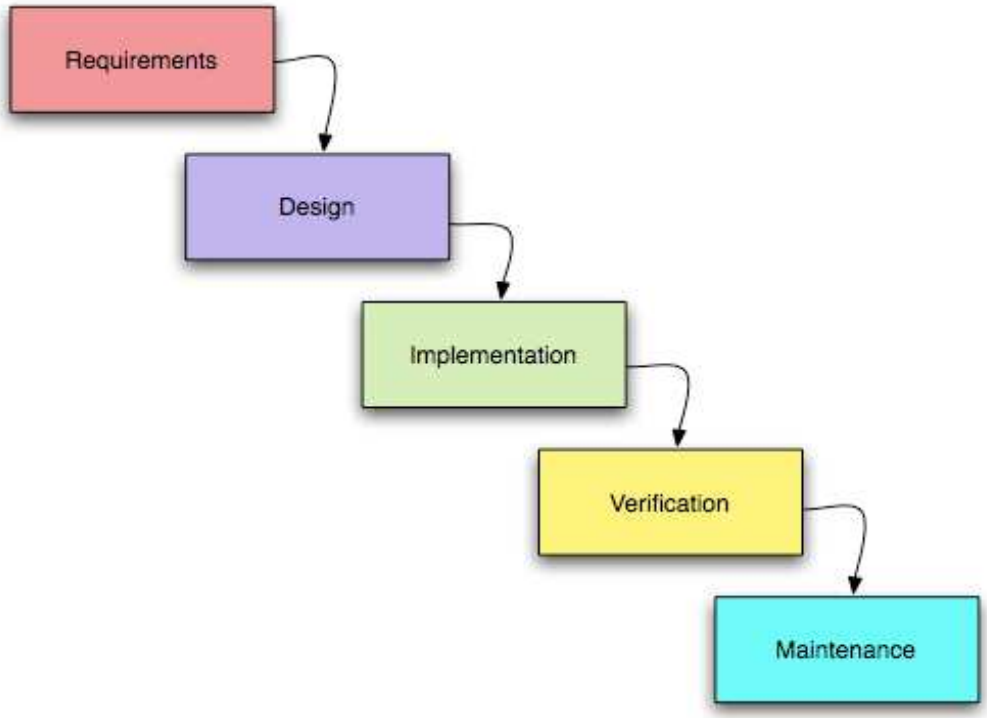
att nämna några exempel). Ett initialt system tas snabbt fram, baserat på väldigt abstrakta krav. Detta system förfinas sedan för att tillfredsställa kundens behov. När dessa uppnås kan systemet levereras till kunden. Ett alternativ är att systemet implementeras på nytt, från början, genom att ett annat, mer strukturerat tillvägagångssätt används (t.ex. ett som följer vattenfallsmodellen).

3. *Component-based software engineering (CBSE)* – Här förutsätts det att delar av systemet redan existerar. Utvecklingsprocessen fokuserar på att integrera de olika delarna, istället för att utveckla allting från början.

Här nedan presenteras fyra olika utvecklingsmetoder som Sharp, et al. (2007) anser vara av stor vikt inom mjukvaruutveckling och som dessutom lyfts fram av Sommerville (2004), Avison och Fitzgerald (2003) samt Apelkrans och Åbom (2001).

2.3.1 Vattenfallsmodellen

Vattenfallsmodellen var den första modellen inom mjukvaruutveckling och utgör grunden för många utvecklingsmetoder som används idag (Sharp et al., 2007). Innan modellen föreslogs år 1970 fanns det inget enat tillvägagångssätt kring hur mjukvara bör utvecklas. Modellen är ett typexempel på en linjär livscykelsmodell i vilken varje steg måste slutföras innan nästa kan påbörjas (se Figur 1). Namnen på de stegen som ingår kan variera men vanligtvis inkluderar modellen fem steg: kravanalys, design, kodning, testning, underhåll (Sharp et al., 2007).



Figur 1: *Vattenfallsmodellen i dess grundutförande. Observera avsaknad av iteration* [1].

Enligt Apelkrans och Åbom (2001) har det snart efter att vattenfallsmodellen blivit känd, påpekats att det inte fungerar med det enkla, linjära, vattenfallsliknande tillvägagångssätt. Som resultat på denna kritik har det uppstått variationer där det förekommer hopp tillbaka till föregående steg (dessa återspeglas *inte* i Figur 1). Denna förändring påpekas även av Sharp et al. (2007), där det nämns att även om iterationen har inkorporerats i vattenfallsmodellen snart efter att den blivit populär, så ingår inte det iterativa tänkandet i modellens filosofi. Feedback som fås från att gå tillbaka till ett tidigare moment i modellen kommer endast från utvärdering från utvecklare, påpekar författarna, och möjligheten till att låta *slutanvändarna* granska och utvärdera systemet är inte inbyggt i den här modellen.

Enligt Bell (2005) lämpas modellen bäst för stora och komplexa projekt. Vattenfallsmodellen bör användas när kraven är väl förstådda och osannolika att förändras under systemutvecklingsprocessen (Sommerville, 2004).

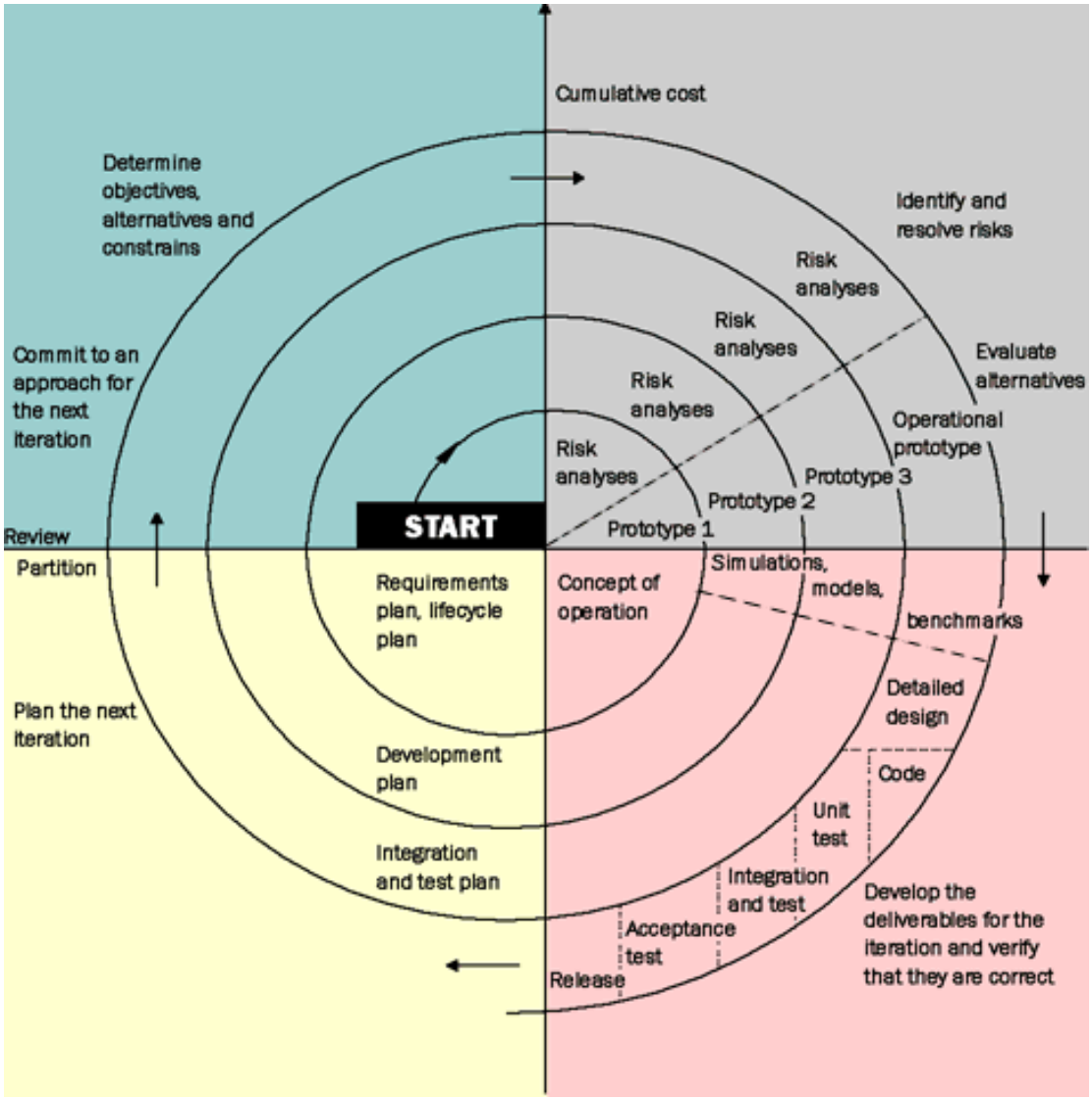
Avison och Fitzgerald (2003) menar att en styrka med vattenfallsmodellen är dess ålder vilket innebär att den är väl använd och beprövad. Enlig Bell (2005) är en av styrkorna med vattenfallsmodellen att den delar upp komplexa uppgifter till mindre och mer hanterbara. Enligt Sommerville (2004) hör vattenfallsmodellens fördelar till att dokumentation tas fram vid varje fas samt att varje fas slutförs innan nästa påbörjas.

En av nackdelarna med vattenfallsmodellen är att ”vattnet” i modellen inte kan rinna uppåt, vilket innebär att tidigare steg under utvecklingen inte kan inte göras om (Bell, 2005). Bell (2005) menar att om det skulle upptäckas något fel som gjorts i tidigare steg finns det ingen möjlighet att gå tillbaka och korrigera det. Ett annat problem med modellen som författaren även nämner är att kraven inte utvärderas förrän produkten är helt färdig och då kan det vara för sent.

Som nackdelar nämner Sommerville (2004) den strikta uppdelningen av projektet till distinkta, nästan åtskilda steg. Förpliktelser måste göras tidigt i processen, vilket gör det svårt att leva upp till kundens förändrade behov och krav. Vidare påstår författaren att eftersom vattenfallsmodellen är så beroende av stabila kravspecifikationer, bör denna modell endast användas där alla inblandade parter är väl införstådda med kraven, och där det är osannolikt att kraven kommer ändras under utvecklingens gång.

2.3.2 Spiralmodellen

Sharp et al. (2007) skriver att vattenfallsmodellen var under många år den dominerande utvecklingsmetoden som utgjorde basen för många projekt. Först år 1988 föreslog Barry Boehm en modell som – på grund av sin utformning (se Figur 2) – har fått namnet spiralmodellen eller Boehms spiral (Apelkrans & Åbom, 2001).



Figur 2: *Spiralmodellen av Boehm (färger har lagts på för att tydliggöra de olika faserna)* [2].

Apelkrans och Åbom (2001) förklarar att utvecklingen börjar i mitten av spiralen, och ju längre avstånd från origo, desto mer tid och pengar har det lagts på projektet. Vidare understryker författarna att i den här modellen tas det hänsyn till risker; inför varje nytt varv i spiralen ställs frågan vilket alternativ som är bäst ur risk- och kostnadssynpunkt.

Även Sommerville (2004) påpekar att den största skillnaden mellan spiralmodellen och andra utvecklingsmodeller är att den förstnämnda lyfter ut riskerna och behandlar dem explicit. Vad som menas med risker är egentligen “allt som kan gå fel”: t.ex. att vissa koncept inte stöds av det valda programmeringsverktyget eller att ett nytt programmeringsspråk som används ännu inte stöds optimalt på den tilltänkta plattformen. Riskerna kan resultera i stora problem för projektet, så som att tidsplanen och/eller budgeten överskrids, och därför är det viktigt att hantera riskerna på sådant sätt att de minimeras.

Spiralmodellen tillämpar iteration genomgående och förenar egenskaper från vattenfallsmodellen och prototyping. Det är just prototyping och den "inbyggda" riskhanteringsmekanismen som lyfts fram av Sharp et al. (2007), där ibid. nämner att till skillnad från vattenfallsmodellen, uppmuntras utvecklarna att ta hänsyn till och reflektera kring alternativa lösningar.

Enligt Boehm (1988) börjar ett projekt som följer spiralen (den gröna delen i Figur 2) med att:

- bestämma målet för projektet (d.v.s. definiera syfte)
- hitta alternativa sätt att uppnå detta mål (d.v.s. definiera alternativ)
- identifiera begränsningar som valet av de olika alternativen leder till (d.v.s. definiera begränsningar)

Sommerville (2004) nämner, förutom de tre punkter, att en detaljerad plan för hantering av projektet upprättas samt att riskerna ska identifieras.

Därefter förflyttas projektet till nästa fas (den gråa delen i Figur 2). Sommerville (2004) poängterar att en detaljerad analys görs för var och en av de riskerna som har identifierats. De olika alternativen och risker utvärderas. Som hjälp vid utvärderingen kan prototyper tas fram. Dessa förfinas och görs mer detaljrika i senare iterationer (d.v.s. längre ut från spiralens origo, i den gråa delen av Figur 2).

Nästa fas innebär att utveckla nästa version av produkten och verifiera att denna är korrekt (den rosa delen i Figur 2). Nu när riskerna är utvärderade kan ett val göras angående en passande utvecklingsmodell för systemet. Exempelvis, om det finns stora risker som har att göra med val av användargränssnitt, kan ett passande angreppssätt vara en iterativ utvecklingsmodell (där många små förbättringar görs och de användartestas ofta). Å andra sidan, om de största riskerna har att göra med systemets säkerhet och tillförlitlighet, kan en modell som är mer strikt och "steg-för-steg" väljas, som t.ex. vattenfallsmodellen (Sommerville, 2004).

Slutligen avslutas ett varv i spiralen i den gula delen av Figur 2, där det är dags att planera för nästa fas. Här tas beslutet om projektet ska fortsätta vidare, och om så är fallet görs planerna för nästa varv upp (Bohem, 1988).

Enligt Apelkrans och Åbom (2001) är spiralmodellen strukturerad och lämpar sig bäst för stora, komplicerade projekt. Modellen "används ofta i forskningsprojekt där många osäkra faktorer skall utvärderas och man är beredd att ta vissa risker" (Apelkrans & Åbom, 2001, s. 51). Boehm (1988) menar på att den första iterationen (*Round 0*) bör ta ca 2 månader, och de efterföljande iterationerna beräknas ta ca 12 månader (egentligen *man-months,* d.v.s. den mängden arbete som utförs av en människa under en månads tid).

Som kritik om spiralmodellen menar Bell (2005) att modellen inte förklarar utförligt hur respektive steg ska utföras, till skillnad från andra modeller.

2.3.3 RAD

Rapid Application Development (RAD) är enligt Sharp et al. (2007) en ny infallsvinkel på mjukvaruutveckling. Begreppet RAD myntades av James Martin år 1991, som ett svar på de problem som de traditionella utvecklingsmetoderna leder till. RAD försöker angripa problemet från en användarcentrerad synpunkt och minimera riskerna som orsakas av ändringar i kravspecifikationen under projektets gång (Sharp et al., 2007). Ett projekt som tillämpar RAD har åtta punkter att följa som riktlinjer. Två av dessa tas upp av Sharp et al. (2007), medan den tredje nämns eftersom den anses vara väsentlig för det här projektet:

- *Time-boxing*: tidsbegränsade cyklar. Enligt vissa bör en timebox inte ta mer tid än sex månader (Sharp et al., 2007), medan andra skriver om 90 dagar (Avison & Fitzgerald, 2003). Vid slutet av varje cykel ska systemet eller en del av det ha levererats. Resultatet av *time-boxing* blir att ett stort projekt bryts ned i flera mindre och att produkten istället för att lanseras för första gången när alla funktioner finns på plats, levereras i mindre paket, där en del av funktionaliteten finns redan under ett tidigt skede i projektet.
- *JAD (Joint Application Development)*: RAD som metod kräver en hög grad av deltagande från alla inblandade i projektet (såväl kunder, användare, utvecklare som beställare) (Avison & Fitzgerald, 2003). Deltagandet uppnås i form av de så kallade *JAD Workshops*, intensiva datainsamlingssessioner där användare, utvecklare och andra intressenter träffas för att diskutera krav som systemet ska uppfylla (Sharp et al., 2007).
- *MoSCoW*-regel: För att överhuvudtaget kunna leverera något inom ramarna för en *timebox* behöver det göras prioriteringar bland de kraven som finns. Som hjälp för att välja ut vad som ska komma med i systemet vid slutet av en specifik *timebox* användes inom RAD den s.k. *MoSCoW*-regeln (Avison & Fitzgerald, 2003). Med hjälp av denna delas kraven upp i:

    - M - *"The must haves"*. Finesser utan vilka projektet inte är genomförbart. Dessa måste finnas med, de utgör minimikraven.
    - S - *"The should haves"*. För att maximera nyttan kommer de här finesserna att tas med, men de är inte kritiska för att projektet ska lyckas.
    - C - *"The could haves"*. Om det finns tid och resurser kommer de här finesserna att tas med, men de kan lätt exkluderas utan att påverka projektet.
    - W - *"The won't haves"*. De här finesserna kommer att exkluderas. De kan - och förmodligen kommer att - tas med i en senare *timebox*, dock inte nödvändigtvis.

Avison och Fitzgerald (2003) fortsätter med att berätta att *MoSCoW*-reglerna är viktiga då de försäkrar att alla krav som upprättas verkligen kan motiveras och att de klassificeras. Vanligtvis tas alla *måste-ha*, åtminstone vissa av *bör-ha* och kanske även några *kan-ha* med i en *timebox*. När det uppstår tidspress däremot kan alla av kraven exkluderas, alternativt skjutas upp till en senare *timebox*, förutom de som ingår i den första gruppen: *must-haves*.

2.3.4 Lättrörlig mjukvaruutveckling

Lättrörlig mjukvaruutveckling (från engelskans *agile software development*) är ett synsätt på mjukvaruutveckling som började formas under det sena 1990-talet (Sharp et al., 2007). *Agile software development* är ett samlingsnamn för flera metoder där grunden utgörs av samma värderingar och principer. Till de mest kända lättrörliga utvecklingsmetoder räknas *Extreme Programming*, *Crystal*, *Adaptive Software Development*, samt *Scrum*.

2.3.4.1 Manifest

Agile Alliance's manifest [6] inleds med påståenden om att rörelsen värdesätter:

- *Individer och samspel* framför processer och verktyg.
- *Körbar programvara* framför omfattande dokumentation.
- *Kundsamarbete* framför kontraktsförhandlingar.
- *Anpassning till förändring* framför att följa en statisk plan.

Manifestet menar att det finns ett värde i alla de saker som räknas upp ovan, men de som står till vänster, i kursivt, värderas högre än de till höger [6].

En del av manifestet är ett antal principer för lättrörliga metoder. De som Sommerville (2004) lyfter fram är bl.a.:

- *Kundinvolvering* – kunden bör vara starkt involverad genom hela utvecklingsprocessen. Kundens roll är att tillhandahålla krav och prioritera vilka som är viktigast, samt att utvärdera varje iteration av systemet.
- *Stegvis överlämnande* – mjukvaran utvecklas stegvis och det är kunden som specificerar vilka kraven som ska inkluderas vid varje iteration.
- *Räkna med förändringar* – utvecklarna ska förvänta sig förändringar av kraven och ska därför designa systemet så att det lätt kan rättas efter de nya förhållanden.
- *Upprätthåll enkelhet* – uvecklarna ska fokusera på enkelhet, både när det gäller mjukvaran som utvecklas och själva utvecklingsprocessen. Där möjligheten finns ska det arbetas aktivt för att eliminera komplexiteten från systemet.

De agila metoderna har blivit oerhört populära under den senaste tiden och fått många anhängare. Trots detta finns det några tankar som kan ses som kritik på de agila metodernas tillvägagångssätt. Sommerville (2004) nämner att:

- Även om idén att involvera kunden i utvecklingsprocessen är attraktiv, beror dess framgång på om kunden är villig och har möjlighet till att spendera tid med utvecklingsteamet, samt om kunden kan representera alla intressenter.
- Utvecklarnas personligheter kan vara olämplig för den intensiva inblandning som krävs och som är typisk för de agila metoderna. Därför kan vissa personer ha problem med att interagera väl med de andra i teamet.

- Prioritering bland kraven kan vara extremt svår, speciellt då systemet har många intressenter.
- Upprätthållande av enkelhet kräver extra arbete och det är något som teamet ofta inte har tid med när en pressad deadline ska hållas.
- Det kan vara svårt att skriva kontrakt för system som utvecklas med de agila metoderna eftersom kraven hela tiden förändras.

Sommerville (2004) påstår att alla metoder har sina begränsningar, och så är även fallet med de lättrörliga metoderna. Enligt författaren anses de agila metoderna vara bäst lämpade för utveckling av små till medelstora system. De lämpas ej för stora projekt, med många utvecklare som befinner sig på olika platser, eller kritiska system, där en utförlig analys av alla kraven är nödvändig för att säkerställa att systemet är säkert och tillförlitligt.

2.3.4.2 Extreme Programming

Enligt Sommerville (2004) är *Extreme Programming* (XP) den mest kända och använda av de lättrörliga utvecklingsmodellerna. Namnet myntades av Beck kring år 2000. Som anledningen till att metoden kallas för "extreme" nämner Sharp et al. (2007) att den utnyttjar en uppsättning av *good practice* extremt. Till exempel anses det vara en *good practice* att testa ofta. Därför är utvecklingen inom XP testdriven och utvecklarna testkör koden flera gånger om dagen. Ett annat exempel på *good practice* är att prata med användare om kraven istället för att skriva ned dem i långa dokument. Därför framställs det – inom ramarna för XP – väldigt lite dokumentation, vilket tvingar fram en dialog med kunden. Som Sharp et al. (2007) säger, är utvecklingen i XP grundat på tester som har baserats på användarnas krav. Underlaget för testerna tas fram med hjälp av olika scenarios som utvecklingsteamet tillsammans med användarna skapar utifrån användarnas krav.

Extreme Programming är perfekt att tillämpa i riskfulla projekt med dynamiska krav [3]. Dessutom passar, enligt Avison och Fitzgerald (2003), XP lämpligast i små- till medelstora projekt där projektet kräver 3 till 10 programmerare. McBreen (2003) säger däremot att XP förmodligen kan lämpa sig på alla typer av projekt och organisationer. Modellen definieras enligt Avison och Fitzgerald (2003) som en mjukvaruutveckling med värderingar av enkelhet, kommunikation, feedback och mod.

Avison och Fitzgerald (2003) säger att trots att XP är en lättrörlig metod så finns det ändå fyra stycken faser i utvecklingsmodellen, nämligen:

1. Planering – en övergripande planering av projektet görs. Denna planering inkluderar prioritering av funktioner, vilka gruppmedlemmar som ansvarar för vad, innehållet av varje iteration.
2. Design – ska utformas med enkelhet, feedback och mod som utgångspunkt, samt möjliggöra för växande förändringar. Under denna fasen är det vanligt att det hålls ett dagligt möte med alla inblandade deltagare.
3. Utveckling – koden utvecklas genom parprogrammering. Testning ska göras med användning av programmeringsverktyg och snabb feedback ska försökas

att få. Extra kod ska utvecklas för att försäkra så att allting fungerar och hela tiden ska koden interageras med den redan existerade implementerade koden.

4. Produktionalisering – ses också som en del av utvecklingen. Skillnaden är att testen i detta steget försäkrar att hela systemet ska anpassas för produktionen, med att exempelvis köra systemet parallellt för att försäkras att systemet fungerar bra. Denna fas övergår sedan naturligt till underhåll av systemet.

Enligt Avison och Fitzgerald (2003) pågår varje iteration i en till tre veckor.

Extreme Programming utgörs av en uppsättning praxis som följs av utvecklarna (Sommerville, 2004). Några av de som nämns av Sommerville (2004) inkluderar:

- Små utgåvor: det minsta möjliga av funktionaliteten som behövs för att produkten ska ha något kommersiellt värde utvecklas först.
- Enkel design: systemets gränssnitt utvecklas endast så mycket som krävs av den nuvarande funktionaliteten.
- Parprogrammering: utvecklarna arbetar i par, kontrollerar varandras arbete och tillhandahåller hjälp.
- Kollektivt ägande: utvecklarna arbetar tillsammans med all kod, så inget enskilt område isoleras. Alla äger allt och kan förändra allt.
- Kontinuerlig integrering: så snart som arbetet med en del av systemet (en s.k. *task*) har avslutats, integreras denna del med resten av systemet. Efter varje sådan integration testas hela systemet.
- Hållbart tempo: mycket övertid anses ej vara godtagbart inom XP då bieffekten blir ofta lägre kvalitet på kod och lägre produktivitet.
- Kunden på plats: en representant för slutanvändaren (oftast kunden) borde vara tillgänglig för utvecklingsteamet under hela utvecklingsprocessen. Inom XP ses kunden som en medlem i utvecklingsteamet och ansvarar för att leverera kraven som utvecklarna ska implementera.

Eftersom leveransen står som fokus i XP, koncentreras inte mycket på dokumentationen, vilket kritiseras av Avison och Fitzgerald (2003).

2.4 Användare

Det finns många tolkningar av vad en användare innebär och det är viktigt att involvera rätt sorts användare (Sharp et al., 2007). Den mest lättbegripliga beskrivningen av en användare är de personer som interagerar direkt med produkten för att utföra en uppgift. Sharp et al. (2007) redogör för tre typer av användare: primära, sekundära och tertiära. De primära användarna är de som frekvent använder systemet, de sekundära användarna innefattar de personer som använder systemet sporadiskt och de tertiära användarna är de som har inflytande på inköpet av systemet eller som kommer att påverkas vid systemets introduktion.

2.5 Krav

För att utvecklare ska få fram en bra kravspecifikation är det viktigt att det läggs ner mycket arbete på denna menar Eriksson (2007). Om det inte skulle läggas stor vikt vid kravhanteringen finns det risk att det kan bli stora felaktigheter i den slutliga kravspecifikationen. Felen kan i sin tur också leda till konsekvenser som att systemet inte blir färdigt i tid, kostnaderna ökar eller att beställaren blir missnöjd och förlorar kunder. Av ovannämnda anledningar är det därför viktigt att genomföra en bra kravhantering från början.

Ett krav är ett påstående som avser en specificerad beskrivning av en produkt, vad den ska göra och hur den ska bete sig (Sharp et al., 2007). Kraven finns i många olika former och ett mål med kraven är att de ska vara så specifika, klara och tydliga som möjligt. Kraven delas oftast in i funktionella och icke-funktionella krav, därutöver finns det även bl.a. data-, miljö- och användarkrav. Enligt Sharp et al. (2007) beskriver funktionella krav vad systemet ska kunna utföra. Sommerville (2004) tillägger att funktionella krav består av påstående över vad systemet ska tillhandahålla, hur det ska reagera vid input och hur det ska bete sig i vissa situationer. De icke-funktionella kraven återger enligt Sharp et al. (2007) vilka egenskaper systemet ska ha. Sommerville (2004) tillskriver att de icke-funktionella kraven innehåller krav som exempelvis innefattar prestanda och tillgänglighet. Datakrav innehåller vilken typ, storlek, omfång, giltighet samt värdet av nödvändig data (Sharp et al., 2007). Användbarhetskraven innefattar bl.a. krav som beskriver svårighetsgraden under inlärningsperioden samt hur säkert och effektivt systemet är.

En del av kravgenereringen är att samla in data. Målet med datainssamlingen är enligt Sharp et al. (2007) att samla in tillräcklig, relevant och lämplig data, så att stablia krav kan produceras. Även om det redan existerar krav kan datainsamling vara lämpligt för att utveckla, tydliggöra och bekräfta de initiala kraven. Generering av alternativ är en huvudprincip i de flesta designutvecklingsmetoderna. Det bästa sättet för att få en bra idé är att generera många idéer och då kan brainstorming, som är en idégenereringsmetod, vara lämplig att använda menar Sharp et al. (2007). Brainstorming är bra då generering, förfining samt utveckling av idéer önskas och är speciellt användbart för framtagning av alternativa lösningar.

Enligt Sharp et al. (2007) är frågeformulär (eller enkät som det även kallas) en väl etablerad teknik för att samla in demografisk data och åsikter. Frågeformulär kan liknas vid intervjuer och frågorna kan vara både öppna som stängda. Det är dessutom av stor vikt att frågorna är väl formulerade så att alla som förfrågas enkelt kan förstå frågorna, även om frågeställaren inte är närvarande. Fördelen med frågeformulär är att de tar fram specifika svar från en bred grupp människor vilket gör det till ett snabbt och billigt sätt att ta fram information på. Vid utformandet av enkäter finns det enligt Sharp et al. (2007) en del aspekter att tänka på:

- Ordningen på frågorna ska vara väl genomtänkt.
- Det ska finnas klara instruktioner på hur enkäten ska genomföras.
- Det ska finnas en balans på enkätens längd, den får inta vara för lång då orkar ingen svara på den.

Kotler, Armstrong, Saunders och Wong (2002) nämner även att det ska finnas en bred skala med svarsalternativ så att rättvisa resultat kan uppnås. Dessutom måste en medvetenhet om frågeordning finnas då utformningen av enkäten annars kan ge upphov till felaktiga resultat.

Enligt Sharp et al. (2007) är en av huvudingredienserna i den användarcentrerade metoden väldefinierade och detaljrikt beskrivna användarprofiler. Ett sätt att skapa en användarprofil på, är genom framtagande av s.k. *personas*. En persona är ett sätt att verkliggöra användarprofilerna genom att beskriva dem på ett detaljrikt och verklighetstroget sätt. En persona beskriver inte verkliga människor utan en sammansättning av de observerade användarnas egenskaper och inkluderar en beskrivning av den låtsades utövarens kunskaper, attityder och omgivning.

Upprättandet av scenarios är ofta det första steget vid framtagandet av krav enligt Sharp et al. (2007). Scenario är en informell berättelse som framställer de mänskliga aktiviteterna eller uppgifterna som tillåter en undersökning och diskussion av omgivning, behov och krav. Fokus i dessa berättelser ligger på att hitta användarnas behov och mål. Genom att skriva enkla berättelser, som är ett naturligt sätt för människor att beskriva vad de gör, kan intressenterna förstå och deltaga i utvecklingsprocessen. Scenarios kan enligt Sommerville (2004) vara särskilt användbara för att lägga till detaljer på övergripande krav. Sharp et al. (2007) nämner att förståelsen av vad användarna utför är en bra startpunkt för att utforska vilka restriktioner, omgivning, irritationsmoment och hjälpmedel användaren arbetar under.

2.6 Prototyper

Prototyper är enligt Sharp et al. (2007) värdefulla att använda på så sätt att de underlättar kommunikationen med intressenter. Dessutom är prototyper effektiva då de hjälper till att stödja utvecklarna att välja bland alternativen samt för att besvara frågor. Det finns två typer av prototyper: low-fidelity- och high-fidelityprototyp.

2.6.1 Low-fidelityprototyp

En low-fidelityprototyp är enligt Sharp et al. (2007) en prototyp som inte liknar slutprodukten särskilt mycket, exempelvis används material som skiljer sig markant från den slutliga versionen. Low-fidelityprototyp innebär att utvecklarna tar fram enklare skisser på hur systemet kan tänkas att se ut och fungera. Low-fidelityprototyperna är användbara eftersom de har som tendens att vara enkla, billiga och lätta att producera. Därtill ses denna typ av prototyping som användbar i det tidiga designarbetet då den tillåter idéer att utvecklas. Low-fidelityprototyper kan tas fram på olika sätt där s.k. indexkort är ett exempel (Sharp et al., 2007). Genom indexkorten kan användaren enkelt interagera med prototypen och varje kort representerar en skärmdump eller en beståndsdel av ett moment. Dessutom är indexkort väldigt effektivt att använda då webbsidor ska utvecklas.

2.6.2 High-fidelityprototyp

En high-fidelityprototyp innehåller mer funktioner än low-fidelityprototyperna och försöker likna den slutliga produkten. Det finns dock en hel del problem med framtagning av en high-fidelityprototyp, som att de t.ex. tar lång tid att bygga, testpersonerna tenderar att ge kritik mot de yttre aspekterna och inte själva huvudinnehållet samt att en mjukvaruprotoptyp kan sätta för höga förväntningar hos slutanvändarna (Sharp et al., 2007).

Vid utveckling av en prototyp måste oftast två avgörande kompromisser handskas med varandra vilka är bred funktionalitet kontra bredd. Dessa benämns *horisontella prototyper*, där en stor variation på funktioner men med få detaljer tillhandahålls. Den andra benämns *vertikala prototyper* och innehåller få funktioner med många detaljer (Sharp et al., 2007).

2.7 Test

Enligt Hjelm (2004) är det svårt att hitta en testmetod som ger en rättvisande indikation av vad användarna tycker och som inte kostar för mycket eller tar för lång tid.

Användbarhetstesting är ett tillvägagångssätt som lägger tonvikt på de egenskaper som är användbara i den utvecklade produkten (Sharp et al., 2007). Testingen är viktigt och görs ofta i ett senare skede av designutvecklingen för att försäkra konsistens vid navigering och hur systemet besvarar användaren. Vid testning bör speciellt konstruerade laboratorium användas med anpassad utrustning menar Sommerville (2004). Dessa testen är väldigt kostsamma och ekonomiskt orealistiska för små projekt med begränsade resurser.

Testing innebär att en mätning av den typiske användaren när den interagerar med produkten görs (Sharp et al., 2007). Samtidigt som uppgiften utförs blir testanvändaren iakttagen, eller observerad som det också kallas, och ibland även inspelad på videofilm. Hjelm (2004) skriver att det är meningsfullt att låta användaren tänka högt då de utför testingen då det ger feedback i form av deras kommentarer. Dessutom kan enkät och intervju vara ett komplement för att frambringa användarens åsikter (Sharp et al., 2007). Intervjuer kan ses som konversationer med ett syfte. Det finns fyra typer av intervjuer där öppna, ostrukturerade intervjuer är en av teknikerna. Denna intervjutyp används för att undersöka åsikter och en fördel är att den genererar en hel del information. Informationen kan dock bli för omfattande att den är svår att analysera och tidskrävande. Vidare beskriver Sharp et al. (2007) att 5-12 användare är ett acceptabelt antal testanvändare men det går att använda färre testpersoner när det finns en liten budget och tidsbegränsningar. Hjelm (2004) menar att det i regel räcker med fem testanvändare. Målet är att testa huruvida produkten är användbar för det syfte som den är utformad för och hur effektiv den är.

2.8 Utvärdering

Enligt Sharp et al. (2007) vill användarna ha produkter som är enkla att lära, effektiva, säkra och tillfredsställande. För att uppnå detta är det viktigt att utvärdera produkten. Utvärderingen är en process där användbarheten och acceptansen hos användarna bestäms. Sommerville (2004) tillägger att vid undersökningen ska det bestämmas huruvida produkten överrensstämmer med de uppsatta kraven. DECIDE är ett ramverk som tillhandahåller en checklista för att enklare kunna planera utvärderingen (Sharp et al., 2007):

Determine – Bestämma de övergripande målen som utvärderingen avser.
Explore – Utforska specifika frågor att besvara.
Choose – Välj utvärderingstillvägagångssätt och metoder för att besvara frågorna.
Identify – Identifiera de praktiska utfärden, som t.ex. välja testpersoner.
Decide – Bestämma hur etiska frågor ska behandlas.
Evaluate – Utvärdera, tolka och presentera data.

Stegen i ramverket är relaterade till varandra och det är därför viktigt att – liksom under hela processen vid framtagandet av prototypen – arbeta iterativt.

Sharp et al. (2007) menar att tidpunkten för utvärderingen beror på vilken produkt eller tjänst som utvecklas.

2.9 Databashantering

Enligt Padron-McCarthy och Risch (2005) är data uppgifter av olika slag. Data kan skiljas från information som är data som tolkats och begreppet kunskap beskrivs som anvisningar om beteende.

Definitionen på en databas lyder enligt Connolly och Begg (2002) som en delad samling logiskt relaterad data, och en beskrivning av denna data, designad för att möta informationsbehoven i organisationen. Författarna förklarar ett databashanteringssystem som ett mjukvarusystem som tillåter användare att definiera, skapa och underhålla data i databasen, och som ger kontrollerad tillgång till databasen. Padron-McCarthy och Risch (2005) säger att en databashanterare oftast är ett stort och komplicerat program.

Connolly och Begg (2002) förklarar att precis som i verkligheten har objekt olika relationer till varandra, där objekten kopplas samman till en logisk helhet genom relationer. Relationsmodellen går ut på att data lagras i relationer skriver Padron-McCarthy och Risch (2005). En relation är samma sak som en tabell, med rader och namngivna kolumner. “En databashanterare som använder sig av relationsmodellen för att lagra data kallas relationsdatabashanterare” (Padron-McCarthy & Risch, 2005, s. 76). När en relationsdatabas designas är målet att upprätta en noggrann representation av data, dess relationer och restriktioner enligt Connolly och Begg (2002). För att uppnå detta måste lämpliga uppsättningar av relationer identifieras. En

teknik som då kan vara användbar för att identifiera relationer kallas normalisering. Normalisering innebär att tillämpa ett antal regler för att uppnå vissa designmål och syftar till att eliminera redundans (dubbellagring). Teorin som Padron-McCarthy och Risch (2005) framställer om normalisering beskriver sex s.k. normalformer där normalformerna är villkor som en tabell kan uppfylla. För att kunna förstå normalformerna finns det ett antal begrepp som behöver redogöras:

- Funktionellt beroende är en relation mellan attribut i en tabell och innebär att om det finns ett värde för ett attribut, exempelvis X, kan värdet av ett annat attribut, exempelvis Y, fås, d.v.s X → Y (Connolly & Begg, 2002).
- Supernyckel ett attribut eller en kombination av attribut vars värden garanterat är unika (Padron-McCarthy & Risch, 2005).
- Kandidatnyckel är en minimal supernyckel där det inte kan tas bort några attribut om den fortfarande ska vara garanterat unik (ibid.).
- Primärnyckel är "ett fält, eller en kombination av fält, som är garanterade att ha unika värden bland alla posterna i tabellen" (Padron-McCarthy & Risch, 2005, s. 416). Dessutom sorteras tabellen som förval efter primärnyckeln. Connolly och Begg (2002) beskriver primärnyckel som den kandidatnyckel som är vald att vara unik och identifiera varje förekomst av en rad i en tabell.
- Främmande nyckel kan även kallas sekundärnyckel och är det attributet som ingår i en relation med en primärnyckel (Padron-McCarthy & Risch, 2005).
- Kompositnyckel: en kandidatnyckel som består av två eller fler attribut (Connolly & Begg, 2002).

Tre av de sex vedertagna normalformerna som nämns av både Connolly och Begg (2002) samt Padron-McCarthy och Risch (2005) beskrivs enligt följande:

1. Första normalformen (1NF): alla värden ska vara atomära, vilket innebär att endast ett värde får förekomma i varje cell.
2. Andra normalformen (2NF): Då 1NF är uppfylld, plus att alla icke-nyckelattribut ska vara fullständigt funktionellt beroende (ffb) av hela primärnyckeln. Alltså måste icke-nyckelattributen vara ffb av båda nyckelattributen i t.ex. en kompositnyckel. Det är då ett tips att dela upp tabellerna i flera tabeller.
3. Tredje normalformen (3NF): Då 2NF är uppfylld, plus att inget nyckelattribut får vara ffb av något annat icke-nyckelattribut.

ER-modellering är ett sätt att kommunicera på ett icke-tekniskt sätt kring vad som menas med data och vilka relationer som förekommer, på ett sätt som gör det möjligt för även av icke-tekniker inblandade i projektet, t.ex. designers, testpersoner, chefer och slutanvändare, att förstå databasen (Connolly & Begg, 2002). ER-modellering är ett exempel på en konceputell datamodell som enligt Padron-McCarthy och Risch (2005) är en av tre klasser av datamodeller. Författarna menar att en konceptuell datamodell är en datamodell där verkligheten beskrivs i termer, men där ingenting sägs om hur datan ska lagras i en databas. Inom ER-modellering ritas de typer av saker som ska finnas med i databasen upp i form av rektangulära lådor. De olika sakerna kallas för *entiteter*. Därefter ritas sambanden mellan *entiteterna* upp i form av

romber. Sambanden kallas för *relationer*. Det finns tre typer av samband som författarna behandlar (ibid.):

- Ett-till-ett-samband (1-1): Exempelvis kan en person endast köra en bil åt gången, och en bil kan endast köras av en person åt gången.
- Ett-till-många-samband (1-N): Till exempel kan en person äga hur många bilar som helst, men en bil kan endast ägas av en person åt gången.
- Många-till-många (N-N): Ett exempel kan vara att en person kan äga flera hus samtidigt, och ett hus kan gemensamt ägas av flera personer.

Ett många-till-många-samband implementeras genom att en kopplingstabell (sambandstabell) upprättas som innehåller de nödvändiga primärnycklarna för att koppla ihop entiteterna (ibid.).

2.10 Design för mobila enheter

Jones och Marsden (2006) skriver att en interaktionsdesigner bör ta reda på – redan innan projektet startat – vad användare verkligen vill ha, vad de inte vill vara utan, vad de vill använda om och om igen, samt vad de vill lägga sin tid och sina pengar på. Hjelm (2004) har tagit fram tips som utvecklarna ska ha i åtanke vid utveckling av mobila enheter. Först och främst säger författaren att en mobilanvändare inte har mycket nytta av bilder och därför ska budskapet främst utformas med text. Jones och Marsden (2006) menar att gränssnittet inte ska missuppfattas med interaktionen, och att gränssnittsdesign skiljer sig från interaktionsdesign. Fokus i gränssnittsdesign är produktens detaljerade utseende och känslan utseendet förmedlar, medan i interaktionsdesignen (som prioriteras högre) identifieras produktens funktioner som ser till att produkten uppnår användarens behov, mål och önskningar (Jones & Marsden, 2006).

För att planera hur gränssnittet kommer att utformas kan en s.k. mock-up göras, som är en modell av ett föremål utan att ha dess funktioner och egenskaper. Mock-up:en kan eventuellt användas för vidare utveckling (Sharp et al., 2007).

Jones och Marsden (2006) tar upp tre huvudaktiviteter för en effektiv interaktion:

- Förstå användare – ha en uppfattning om användarnas kompetenser och begränsningar, få en bild av hur de lever samt vad de gör och använder.
- Utveckla prototyper – presentera en tänkt interaktionsdesign så att den kan demonstreras, ändras och diskuteras.
- Utvärdering – utvärderingstekniker identifierar styrkor och svagheter av designen.

Ett annat problem för mobilanvändare är webbplatser som är beroende av att användaren integrerar med dem via mus, menar både Paelke, Reimann, och Rosenbach (2003) och Hjelm (2004). “Om sidorna är fulla av menyer som skall kommas åt via musen” kommer det att bli mycket svårt för en mobil användare att integrera med innehållet (Hjelm, 2004, s. 43). Dessutom kan användarens

inställningar inte styras och därför bör webplatsen utformas så att den är så lättillgänglig som möjligt. Designers bör tänka på att det inte är de som bestämmer hur webbplatsen ser ut när den kommer till mobilen och därför bör de tänka i termer av användarupplevelser och inte i termer av grafiska designelement. Nästa tips berör färgblindhet. Hjelm (2004) menar att sidan bör skrivas ut i svartvitt (med gråskalor) för att på så sätt få en inblick av de färgblindas uppfattning av sidan och för att framställa kontrasterna så att de är riktigt tydliga. Därtill ska länktexterna vara tydliga med en beskrivande text som också är länkad t.ex. "För mer information om kostråd, klicka här". Vanlig understruken text bör undvikas på sidor, nämner författaren, eftersom det är lätt, särskilt på en mobilskärm, att blanda ihop den med en länk.

Genom användning av *Cascading Style Sheets* (CSS) kan överföringstid och därmed pengar sparas skriver Hjelm (2004). Författaren menar att det mest effektiva sättet för att åstadkomma mindre överföringstid är att använda en formatväljare i CSS som anpassar innehållet efter plattformen. Detta innebär att bara den aktuella formatmallen för presentationformen kommer att hämtas.

Hjelm (2004) skriver att bilder är en form av grafik och de bör ha ett samband med den text de illustrerar, ha en funktion, vara relevanta och förnuftiga. Därtill måste bilderna vara små, för att vara snabba att överföra. Paelke et al. (2003) menar att endast ett begränsat antal färger bör användas då många mobila enheter oftast bara kan visa ett begränsat antal kulörer på displayen. Hjelm (2004) menar att bilder även måste vara enhetliga i både stil och storlek. Något som är bra att komma ihåg enligt författaren är att en bild tar bandbredd och tid att hämta in till mobilen, medan en färg kan genereras direkt av webbläsaren.

Om formulär från mobila terminaler ska användas gäller den gyllene regeln "ju mindre, desto bättre", främst på grund av de begränsade inmatningsmöjligheterna (Hjelm, 2004, s. 165).

2.11 Webbdesign

Tidigare var webbsidor mestadels textbaserade sidor som tillhandahöll hyperlänkar till olika webbplatser eller sidor som bestod av text (Sharp et al., 2007). Stor ansträngning gick åt att planera ut på vilket sätt informationen skulle struktureras upp på i gränssnittet, för att möjliggöra för användarna att navigera och få tillgång till en plats snabbt och enkelt.

Sharp et al. (2007) berättar att Jakob Nielsen anpassade sina och Ralph Molichs användarriktlinjer för att tillämpa de för webbdesign. Fokus låg på enkelhet, feedback, hastighet, tydlighet och bekvämlighet. Nielsen [4] påpekar att nedladdningstiden för en webbplats är viktigt att tänka på, mestadels på grund av att användare söker sig till annat om de får vänta för länge på att få se en sida. För att undvika en lång nedladdningstid rekommenderar Nielsen att minska grafikinnehållet på webbsidan [4]. Genom att låta användarna se innehållet men ändå minska nedladdningstiden så kan stora bilder göras om till små och länka de till de stora, eller länka till bilderna från hypertext. Andra interaktionsdesigners håller inte alltid med Nielsens teori (Sharp et al. 2007). De menar att både en estetiskt behaglig och

samtidigt användbar webbsida kan skapas. Huvudorsaken till att lägga stor vikt på grafiken är för att göra webbsidan mer utmärkande, effektfull och behaglig för användaren när de för första gången ser sidan, samt att få användaren att komma ihåg den till nästa besök (Sharp et al., 2007).

Under det tidiga 2000-talet expanderade webbdesignen med hjälp av olika verktyg och programmeringsspråk som tillhandahöll möjligheter för både designers och vanliga användare att utveckla webbsidor i en mer multimedial miljö (Sharp et al., 2007).

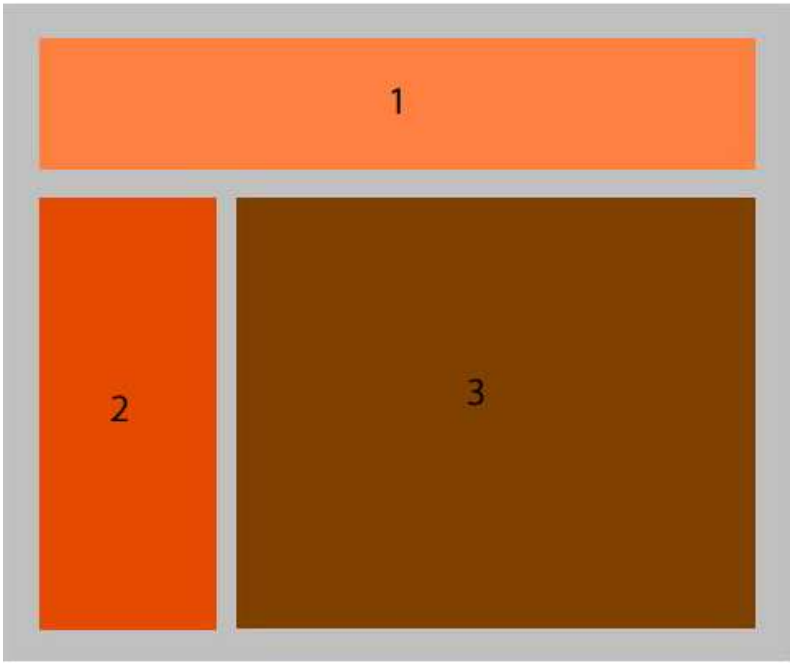
Sharp et al. (2007) redogör för en debatt som hållits, om att webbdesigners skapar webbsidor med tankesättet att användaren kommer att besluta vilken handling de ska utföra utifrån att designen är korrekt enligt webbdesignregler, hur utformningen av text och bild är använt, vilka färger och ikoner som tillhandahålls på sidan o.s.v. Användare tänker inte riktigt likadant utan beter sig annorlunda. Då de besöker en sida ger de en snabb överskådlig blick och skummar igenom delar av sidan, därefter klickar användaren på den första länken som drar till sig deras uppmärksamhet eller som verkar vara den som leder fram till det som eftersöks.

I dagsläget är det viktigt att tänka på att designa bra, presentera och strukturera informationen och tänka på systemets skick [4]. Dagens webbsidor liknar mycket vanliga Graphical User Interfaces (GUI), grafiska användargränssnitt, men den stora skillnaden mellan GUI och en webbsida är att en webbsida innehåller hypertext och navigerar med hjälp av dessa.

Veen (2000) presenterar tre nyckelfrågor en användare ska kunna besvara när den kommer in på en webbsida och som en designer bör tänka på när utformningen på sidan sker:

1. Var är jag? – Användaren kan komma från varsomhelst in på sidan och det är väldigt viktigt att det redan från början syns tydligt vart användaren befinner sig.
2. Vart kan jag gå? – Härifrån ska användaren kunna navigera sig fram till resten av webbplatsen för att se vad som finns mer.
3. Vad finns här? – Den egentliga informationen finns här, och är den största orsaken till att användaren kom till sidan.

Ett sätt att strukturera upp de tre ovannämnda delarna presenteras i Sharp et al. (2007) och ser ut som i figur 3 nedan.



***Figur 3** : Ett förslag till layout för områden som besvarar Veens nyckelfrågor* (Sharp, et al., 2007).

Nielsen [4] menar att det är bättre att använda Cascading Style Sheets (CSS) än ramar när en webbplats skapas. Hjelm (2004) skriver att CSS fungerar som en formatmall där utformningen för alla sidor på en webbplats bestäms, utan att samma kod behöver skrivas på varje enskild sida.

Sharp et al. (2007) beskriver att innehållet på en webbsida bör utformas annorlunda än hur standarddokument görs, på grund av att sättet en användare läser webbsidan på är annorlunda. Innehållet ska vara kort, exakt och lätt att skumma igenom. För att öka chanserna för att användaren ska hitta det som vill förmedlas i sidan, bör den informationen delas upp efter rubriker som fångar in huvudinnehållet. Informationen kan delas upp i preciserade områden för att klargöra ännu mer för användaren [4].

Hjelm (2004) påpekar att en god tumregel är att inte fråga användaren efter uppgifter på webbplatsen som inte kommer att användas. Vilket innebär att det inte är mycket till nytta att ha en databas full med uppgifter på folk som varit inne på en webbplats om dessa inte används till något.

2.12 Verktyg

ASP.NET är ett ramverk för utveckling av webbapplikationer (Evjen, Hanselman, Muhammad, Sivakumar & Rader, 2006). ASP.NET version 2.0 innebär enligt författarna en dramatisk förändring gentemot de äldre versionerna 1.0/1.1. Tre koncept som är nya för ASP.NET 2.0 är `master pages`, `site navigation` och `membership`.

`Master pages` är ett enkelt sätt att skapa en mall som flera andra sidor baseras på (Evjen et al., 2006). Darie och Watson (2006) tillägger att genom användning av `master pages` kan utvecklaren försäkras att ett konsistent visuellt utseende och funktionalitet kommer att appliceras på alla sidor som utgör en webbplats.

Evjen et al. (2006) skriver att då webbplatser ofta består av många olika webbsidor uppstår ett behov av att länka dem samman, d.v.s. varje sida måste innehålla hyperlänkar till de andra sidorna på webbplatsen. Problem uppstår när utvecklare behöver flytta/döpa om filer, eller på något annat sätt förändra strukturen för webbplatsen. Att gå in i koden för varje sida och för hand ändra länkarna till alla de andra sidorna är något som borde elimineras då det kan vara mycket tidskrävande. Författarna poängterar att ASP.NET 2.0 löser problemet genom introduktion av ett navigationssystem för webbplatsen, där hela webbplatsens struktur definieras i en – och bara en – XML-fil. Därefter kopplas XML-filen (en s.k. `site map`) till ett antal navigationskontroller. Två av dessa kontroller kallas för `SiteMapPath` och `TreeView`. Den förstnämnda finns till för att användaren lätt ska kunna se var denne befinner sig för tillfället, medan den andra ger ett trädlikt vy där alla sidor som utgör webbplatsen syns i en strukturerad hierarki (ibid.).

Många webbplatser kräver att användaren loggar in för att utnyttja funktionerna fullt ut. Eftersom autentisering av användare, enligt Evjen et al. (2006), är så vanligt förekommande i webbutvecklingssammanhang, har Microsoft underlättat arbetet med den genom att introducera den s.k. `Membership and role service` i ASP.NET 2.0. Ramverket är enligt författarna enkelt att implementera och använder Microsoft SQL Server för att lagra data. Ramverket tar hand om inloggning, autentisering, attest (`authorization`) och hantering av de användare som begär tillgång till webbapplikationer. Dessutom tillhandahålls ett antal kontroller, skapade för att användas på webbsidorna, vars syfte är att hantera bl.a. inloggning, skapande av nya användare, ändring av lösenord, återskapande av glömda lösenord, sessionshantering, samt andra aspekter som ingår i ramverket. Evjen et al. (2006) menar att en stor fördel med att använda ramverket, gentemot att själv skriva all denna funktionalitet som utvecklare, är att de komponenter som ingår i ramverket är redan mycket vältestade och tillhandahåller därför mycket hög säkerhet och troligtvis få buggar.

AJAX är en utvecklingsteknik som möjliggör skapande av interaktiva webbapplikationer [7]. AJAX i sig är ingen teknologi utan en teknik som går ut på att utnyttja flera existerande teknologier (så som XHTML, CSS, DOM och XMLHttpRequest). Enligt Microsoft [8] är *ASP.NET AJAX* en utökning av ASP.NET som möjliggör enkel åtkomst till AJAX-tekniken. Utökningen släpptes av Microsoft i slutet av januari 2007, och företaget har fastställt att ASP.NET AJAX kommer att inkluderas i senare versioner av .NET-ramverket.

2.13 Prissättning

Kotler et al. (2002) skriver att det pris som väljs att ta för en produkt påverkar både interna och externa faktorer. De interna faktorerna är sådant som företaget själva kan påverka som t.ex. marknadsföringsmål, val av strategi, kostnaderna och

organisationen som sådan. De externa faktorerna är de som organisationen inte kan styra över vilket bl.a. innefattar konkurrensen, marknaden och efterfrågan. Innan prissättning sker ska ett övergripande mål över produkten bestämmas. Det finns tre olika sorters prissättingsmetoder (ibid.):

- *Kostnadsbaserad* – utgångspunkten innebär att fokus ligger på kostnaden och utifrån denna läggs en standardmarginal på.
- *Värdebaserad prissättning* – priset sätts utifrån kundernas uppfattning om produktens värde.
- *Konkurrensbaserad prissättning* – priset bestäms efter vad kunderna bedömer med konkurrenternas priser som utgångspunkt.

3 Rapport

3.1 Verksamhetsbeskrivning

Media IT är en forskningsgrupp inom Högskolan i Halmstad som fokuserar på tre viktiga förändringar i samhället:

- utvecklingen av nya informationsteknologier
- förändringar i hur media konsumeras
- pågående sammankoppling mellan olika typer av media

Dessa förändringar ger nya, intressanta framtida möjligheter för till exempel digital media inom tidningsbranschen. Den nya tekniken, såsom mobila IT-enheter och e-papper, skapar behov av nytänkande inom områden som gränssnittsdesign, interaktionsdesign och applikationsutformning. Media IT-gruppens studier fokuserar på design av användbara medietjänster för den nya tekniken.

Syftet med Media IT är att ta ett helhetsgrepp på alla digitala kanaler såsom webb, mobil och e-papper; där har gruppen valt att arbeta med ett projekt som de kallar för UbiMedia. Den övergripande forskningsfrågan för projektet är:

> *”How can ubiquitous information environments be designed to support production and distribution of profitable ubiquitous media services leveraging user value?”*

3.2 Systemeringsprocess

3.2.1 Diskussion kring utvecklingsmetoder

Enligt Eklund (2002) gäller det vid projektarbete att ha en underliggande modell. Av de tre grupper av livscykelsmodeller som presenteras av Sommerville (2004), kan

gruppen konstatera att endast två är relevanta för vidare granskning, nämligen *the waterfall approach* och *evolutionary Development*. Den tredje gruppen, CBSE, kan inte tillämpas i det här projektet, då metoden förutsätter att det redan finns färdiga delar av det systemet som vidareutvecklas och förfinas.

Vattenfallsmodellen som är en välbeprövad modell enligt Bell (2005), lämpar sig bäst i stora komplexa projekt. Dessutom, som Sommerville (2004) säger, krävs det att det finns väldefinierade och oföränderliga krav eftersom det inte i efterhand går att hoppa tillbaka till föregående steg enligt Bell (2005). På grund av ovannämnda anledningar har vi valt att inte arbeta efter denna modell främst då vi räknar med att våra krav kommer att förändras under projektets gång och kan inte antas vara tillräckligt stabila, då gruppen inte fått några fasta riktlinjer från beställaren. Dessutom anses inte modellen som tillämpbar, då gruppen saknar tidigare erfarenheter av denna typ av projekt, och då kan det vara förödande att använda vattenfallsmodellen eftersom modellen, som Bell (2005) påpekar, inte upprepar tidigare slutförda steg. Även om modellen ej tillämpats fullt ut, har vissa delar valts ut och kommer att tillämpas i vårt utvecklingsarbete.

Spiralmodellen som enligt Apelkrans och Åbom (2001) lämpar sig bäst för stora, komplexa projekt kan anses vara för omfattande för att tillämpa i vårt utvecklingsarbete. Bohem (1988) menar att den första iterationen bör ta två månader och de efterföljande beräknas ta tolv månader. Både Andersen et al. (1994) och Eklund (2002) skriver att ett projekt definieras som avgränsat i tid och omfattning. Detta stärker våra argument om att inte tillämpa spiralmodellen.

Enligt Avison och Fitzgerald (2003) är XP en modell som lämpar sig väl för små och medelstora projekt med tre till tio programmerare. Extreme Programming är utmärkt att använda i riskfyllda projekt med dynamiska krav [3], vilket passar väl in på vårt projekt. Därutöver strävar gruppen efter att tillämpa Avison och Fitzgeralds (2003) definition på XP som en mjukvaruutvecklingsmetod som högt värderar enkelhet, kommunikation, feedback och mod.

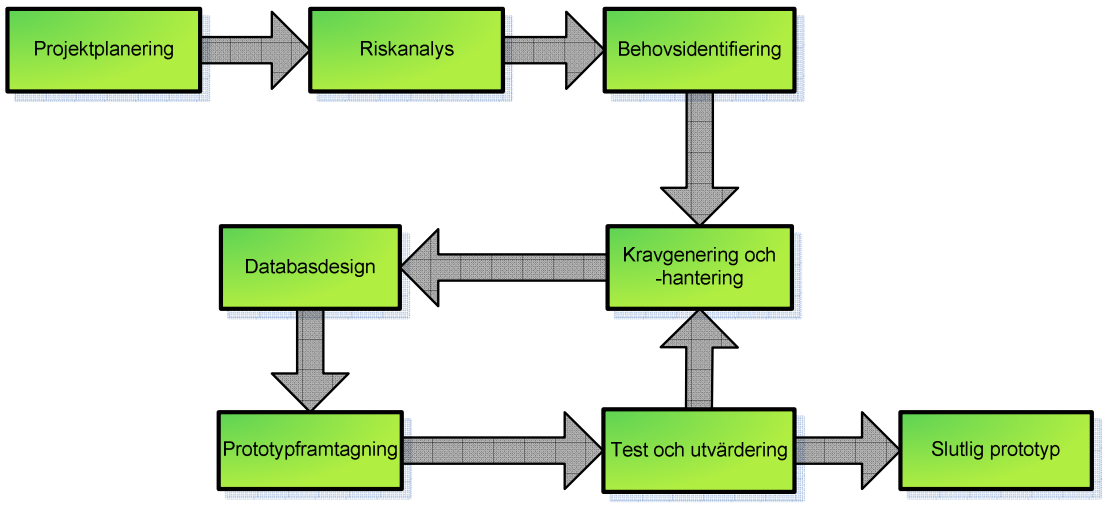
Extreme Programming har inte följts till punkt och pricka utan har använts som en övergripande modell vid projektets gång. Sharp et al. (2007) menar att alla som behöver använda sig av en utvecklingsmetod kommer att behöva lägga till specifika detaljer gällande omständigheterna. Gruppen har arbetat efter föregående påstående och tillämpat diverse modeller som vävts in efter behov, vilket gjorts för att möjliggöra utvecklingsarbetet helt och hållet. Enligt både Sharp et al. (2007) och Avison och Fitzgerald (2003) framställs det väldigt lite dokumentation när XP används som utvecklingsmetod. Projektgruppen har haft detta i åtanke under projektets gång och därmed arbetat hårt med att dokumentera det som åstadkommits. Dokumentationsdelen är av stor vikt vid arbete med vattenfallsmodellen enligt Sommerville (2004). Därför skulle denna del kunna vävas in i vårt utvecklingsarbete då löpande dokumentering har gjorts genomgående under projektets gång.

Delar av Rapid Application Development (RAD) har använts under utvecklingsprocessen. Ett exempel är time-boxing, om än i en mycket modifierad form, då utvecklingscykeln har fått förkortas från de 6 månader som Sommerville (2004) föreskriver, eller Avison och Fitzgeralds (2003) rekommendation om 90 dagar till så korta cyklar som femdagarsperioder. Anledningen till detta är projektets totala

längd på tio veckor, där utvecklingsprocessen utgör endast en del av tiden, tillsammans med bland annat teoretiska studier och undersökningar. RAD har även tillämpats under prioritering av kraven, vilket har gjorts efter den metod som går under benämningen MoSCoW-regeln och beskrivs av Avison och Fitzgerald (2003) förklarar. Det skulle även vara önskvärt för vårt projekt att under projektarbetet använda Joint Application Development, som enligt Avison och Fitzgerald (2003) kräver ett högt deltagande av alla inblandade parter där dessa träffas och diskuterar kraven som systemet ska uppfylla. På grund av den knappa tiden projektet omfattar samt intressenternas tidsbrist har det inte funnits möjlighet till denna form av diskussioner.

3.2.2 Tillvägagångssätt

Gruppen har under projektets gång tillämpat diverse delar från de beskrivna modellerna i teoriavsnittet. Här nedan sammanfattas i en kort beskrivning de stegen som genomförts under projektets gång, vilka även sammanfattas i figur 4.



***Figur 4**: Utvecklingsmodell för projektet*

Innan projektet satte igång upprättade gruppen en projektplan (se bilaga 1). Projektplanen beskriver vad som ska göras under projektets gång, beställarens verksamhet, arbetsroller, tidsplan, verktyg, etc. Precis som Eklund (2002) säger är det viktigt att skriva en projektplan för att bl.a. klargöra ansvarsområden, konkretisera tidsplaner och beskriva projektets mål så att alla i projektgruppen har en övergripande plan att följa. Detta steg kan jämföras med fas ett i spiralmodellen som Boehm (1988) beskriver på så sätt att mål och alternativ för att uppnå dessa ska upprättas. Förutom detta ska det upprättas en övergripande planering under XP:s första steg, menar Avison och Fitzgerald (2003). Då vår utformade projektplan var färdigställd skulle denne sändas in till handledaren för ett godkännande. Dock ansågs tidsplaneringen inte fullständig så att en ny, i form av Ganttschema, fick utformas (se bilaga 2).

Inledningsvis startades projektet upp med att utforma en riskanalys där alla tänkbara risker som kunde vara hotande för projektet identifierades. Riskerna analyserades och bedömdes efter hur stor sannolikhet det är att de kan inträffa. Riskanalysen kan liknas vid spiralmodellens fas två där Somerville (2004) poängterar att risker ska identifieras och analyseras.

Därefter tog det första riktiga steget vid, vilket innebar att hitta behov och fastställa krav. Detta ingår i vattenfallsmodellens första steg.

Vidare har designfasen påbörjats och utifrån de genererade kraven har databasens entiteter och relationer samt low-fidelityprototyper utformats. Dessa low-fidelityprototyper har varit överskådliga skisser på hur systemet ska se ut. Vattenfallsmodellens steg två innefattar design precis som XP:s andra steg, design. Avison och Fitzgerald (2003) beskriver att designen i XP ska frambringas med bl.a. enkelhet och feedback i åtanke. Föregående påstående har gruppen tagit hänsyn till vid skapandet.

För att få klart för oss huruvida low-fidelityskisserna accepterades av användarna valdes tio testpersoner ut. Personerna fick se på tre framtagna prototyper samt svara på frågor som vi ställde (se bilaga 9 och bilaga 10). Med testpersonernas respons som utgångspunkt kunde projektgruppen fatta ett beslut om att avgränsa kraven. Testning ingår i det tredje steget i XP (Avison & Fitzgerald, 2003). Därefter fortsatte designarbetet och arbetet med high-fidelityprototyper påbörjades. Avison och Fitzgerald (2003) förespråkar att under designfasen ska dagliga möten hållas. Genomgående under projektet har gruppmedlemmarna arbetat i samma miljö och har därför kunnat hålla dagliga sammanträden.

Nästa steg innebar att designa en databas. Enligt Connolly och Begg (2002) är en ER-modell är ett sätt att framställa data och dess relationer på ett icke-tekniskt sätt. Projektgruppen tog fram skisser i form av ER-modeller och designandet kunde påbörjas. Gruppen fick avgränsa ER-modellen för att den annars vore för omfattande för detta projektet (se bilaga 12). Fortsättningsvis har high-fidelityprototypen vidareutvecklats. Därefter genomfördes ett nytt test samt utvärdering på fem testanvändare. För att leva upp till de nya kraven som uppstod vid testet samt utvärderingen fick en uppdatering av databasen göras. Slutligen implementerades databasen utan att vi stötte på några större svårigheter. Informationen som framkom från testet samt utvärderingen har tillvaratagits vid fortsatt förbättring av high-fidelityprototypen.

Enligt Sommerville (2004) värderar vattenfallsmodellen högt att dokumentation sker löpande. Utifrån detta uttalande har vi dokumenterat genomgående under arbetsprocessen. Däremot menar Avison och Fitzgerald (2003) att XP koncentrerar sig endast på att leverera en färdig produkt och inte på dokumentationen. Detta har gruppen varit mycket noggranna med att inte tillämpa utan har som sagt dokumenterat fortlöpande.

Nedan presenteras mer detaljerat vad som utförts under respektive steg.

3.2.3 Projektarbete

Eklund (2002) beskriver hur en projektgrupp växer samman då ett samspel frodas. Vi kan själva se vår grupp som sammanväxt då gruppmedlemmarna tidigare har arbetat gemensamt i olika projekt och har ett bra samspel. Andersen et al. (1994) samt Eklund (2002) definierar ett projekt som att det är engångskaraktäristiskt, målinriktat, avgränsat i tid och omfattning, tilldelat begränsade resurser och uppdelat i delmål. Samtliga av dessa påståenden passar väl in på den projekttypen vi arbetar under.

Carlsson och Nilsson (2002) påstår att projektgrupper bör ha en projektledare. Gruppen har inte haft någon utsedd projektledare; istället har gruppmedlemmarna själva ansvarat för att respektive uppgift blivit genomförd på utsatt tidpunkt. Enligt Carlsson och Nilsson (2002) ansvarar projektledaren för att leda projektgruppen och styra den mot det uppsatta målet. Det kan vara både positivt som negativt att vår grupp saknar projektledare. Negativt på det sättet att det inte finns någon diciplin hos gruppmedlemmarna då de inte har någon som kräver resultat och stämmer av deras arbete, vilket kan leda till att arbetet inte blir utfört enligt tidsplanen. Dock har de möten med handledaren resulterat i att produktionen har hållts igång och regelbundna avstämningar gjorts, däribland har veckorapporter (se bilaga 3) sändts in samt återkommande handledningsmöten har hållts. Det som kan vara positivt med att gruppen saknar projektledare är att det kan skapas utrymme för diskussioner och egna initiativ samt ge upphov till kreativitet.

Eklund (2002) redogör för teorier om att olika individers egenskaper bidrar till dynamik och goda resultat då kunskaperna kompletterar varandre. Detta uttalande håller vi med om till punkt och pricka då gruppmedlemmarna besitter olika kunskaper och erfarenheter. Då vi tidigare arbetat med varandra i projekt vet alla vilka styrkor de andra medlemmarna besitter. På så sätt kan en uppdelning av respektive ansvarsområde komma att bli ganska given. McManus & Wood-Harper (2003) förespråkar att konflikter ska lösas med öppna diskussioner, de få meningsskillnader som uppstått under projektets gång har lösts på detta sätt. Gruppmedlemmarna har dessutom arbetat nära varandra fysiskt och därigenom kunnat föra dialog under arbetets gång vilket har inneburit att arbetsdagen naturligt inletts med små diskussioner för vem som är ansvarig för respektive del.

Eklund (2002) nämner att det är viktigt att vid projektarbete ha en underliggande modell för att undvika förlorad kontroll. Gruppen upprättade innan projektets gång en s.k. projektplan (se bilaga 1). Projektplanen innehåller punkter som berör övergripande beskrivningar över vad och hur gruppen bör tänka på och agera under projekttidens gång. I projektbeskrivningen sammanställdes även en tidsplan där aktiviter staplades upp dagsvis. Enligt vår handledare var inte denna tidsplanering fullständig så att gruppen utformade en ny planering i form av ett Ganttschema (se bilaga 2). I det nya Ganttschemat staplades aktivitetarna upp veckovis, och det visade sig ganska snabbt att tidsplanen var felaktig. Detta kan bero på att gruppen inte besitter några erfarenheter av projekt av denna storlek, och därför är har det varit svårt att uppskatta tiden, omfånget och ordningsföljden på aktiviteterna.

3.2.4 Riskanalys

Utifrån Marttala och Karlssons (1999) beskrivning om riskidentifiering har gruppen tagit fram risker som kan komma att påverka projektet i negativ rikning. Vi har, precis som författaren poängterar, staplat upp riskerna i form av en lista och bedömt hur stor sannolikhet det är att de kan inträffa (se tabell 1). Vi har tagit hänsyn till Sommervilles (2004) uppdelning av risker inom de tre nämnda områden – projektrisker, produktrisker och företagsrisker. Tillsammans i kategorin företagsrisker har vi skrivit grupprisker, just för att vi inte är ett företag utan en projektgrupp. I de olika kategorierna har vi identifierat vilka potentiella risker området kan råka utför samt bedömt möjligheten till risk genom att sätta siffrorna 1 till 10, där 1 innebär att risken är låg att händelsen inträffar, medan 10 betyder det motsatta.

	Bedömning av riskmöjlighet
Projektrisker	
Långsamma åtgärder från handledare	8
Datorfel	5
Felbedömning av tidsplanen	6
Felprioritering (lägger för mycket tid på fel saker)	8
Produktrisker	
Brist på kunskap inom teknik	8
Ej nödvändiga rättigheter	5
Feltolkning av uppgift	5
Oklar information från beställaren	4
Begränsad tid	8
Företagsrisker/Grupprisk	
Konflikter	2
Sjukdom	4

Tabell 1: *Riskanalys*

Åtgärder som vi i gruppen kan göra för att undvika risker är först och främst att följa en bra utformad och genomtänkt planering. Om planeringen hade varit dåligt utformad så hade projektet varit riskfyllt och vi hade stött på fler risker än om vi hade följt en bra utformad och genomtänkt planering. Riskanalysen i sig själv är bra för att säkra det framtida jobbet genom att förebygga risken innan den hinner uppstå.

Det är inte väsentligt för vår grupp att vara alltför djupgående när det gäller att lokalisera kostnader för risk och riskhantering. Detta är på grund av att det inte förekommer några finansiella kostnader i vårt skolprojekt, förutom våra personliga kostnader för studiematerial, kost och logi. En annan kostnad kan vara i form av tid,

men då vi har en fast deadline för projektet kan vi inte spendera mer tid än vad som tillgivits på att slutföra projektet.

3.2.5 Användare

Vi har undersökt och diskuterat vilka användare som skulle utgöra målgruppen för vår tjänst och kommit fram till att våra användare är alla som känner att de behöver stöd med att förbättra sin hälsa. De personer som kommer att använda vår tjänst omfattar inte någon speciell åldersgrupp, utan tjänsten är tänkt vara anpassad för alla åldrar. Däremot har gruppen valt att sätta en åldersgräns på 16 år; detta främst på grund av att det inte anses vara lämpligt att träna på gym i yngre ålder. Denna begränsning togs fram efter att gruppmedlemmarna ringt runt till tre olika gym och konstaterade att alla hade 16 år som åldersgräns. Det som Sharp et al. (2007) benämner primära användare har identifierats vara personer som känner att de vill ha stöd i deras dagliga liv och få en uppmuntran att äta rätt samt motionera, och på så sätt uppnå en bättre hälsa. Sharp et al. (2007) benämner de sekundära användare som de personer som använder produkten tillfälligt och de tertiära användarna som de som kommer att ha inflytande vid produktens inköp. Vi anser att de sekundära användarna för vår tjänst är de som inte behöver stöd i deras kost eller träning utan är medlemmar för att emellanåt kunna läsa artiklar, få recept eller kunna utbyta tankar med andra användare. För tillfället kan inte de tertiära användarna identifieras eftersom tjänsten inte har utvecklats till en tillräckligt hög nivå, för att gruppen ska kunna fastställa vilka de tertiära användarne är, eller kommer att bli.

3.2.6 Krav

Utvecklingsgruppen har endast fått ett initialt krav från beställaren vilket var att utforma en tjänst till både mobila enheter samt webben. Sharp et al. (2007) redogör för ett par olika kravidentifieringstekniker som bl.a. innefattar brainstorming, enkät, intervju, personas samt senarios. Dessa teorier har vi arbetat efter då vi identifierat behov. Utifrån behoven har vi sedan kunnat generera och fastställa krav över vad systemet ska kunna göra.

Sharp et al. (2007) rekommenderar att använda brainstorming för att generera idéer. Då vi från början inte fått specifika anvisningar från Media-IT, på vilken av de olika kolltjänster de ville beställa, började vi med en brainstorming för att frambringa så många förslag som möjligt. Där satte vi oss ner och bollade idéer över vilka funktioner som vi tyckte skulle vara passande under respektive kolltjänst (se bilaga 4).

Nästa steg innebar att vi utformade en enkät (se bilaga 5) och detta gjordes med hjälp av några av de teorier om utformning av enkät som Sharp et al. (2007) samt Kotler et al. (2002) redovisar. Enkäten delades ut till 29 deltagare med ett åldersintervall på 20-60 år. Av enkäten strukturerads resultatet upp (se bilaga 6) och där framgick det att Familjekoll och Skolkoll var den mest eftertraktade tjänsten, tätt följt av Nöjeskoll. Hälsokoll kom på sistaplats med ett medelvärde på 2,8 av 5 möjliga. Vi såg dock ett mönster med enkätens utformning där Kolltjänsternas placering kom i samma ordning

som de låg i enkätens ordningsföljd, vilket gruppen i efterhand kan analysera utifrån frågereglerna Sharp et al. (2007) benämner:

- Ordningen på frågorna ska vara väl genomtänkt
- Det ska finnas klara instruktioner på hur enkäten ska genomföras
- Det ska finnas en balans på enkätens längd, den får inta vara för lång då orkar ingen svara på den.

Utifrån ovannämnda punkter har vi insett att vi inte tillämpat dessa då utformning av enkäten gjordes. Genom att inte följa ovannämnda frågeregler har vi analyserat svaren från enkäten och kommit till konsensus att resultatet påverkats av enkätens längd och frågeordning. Vid senare förfrågning av de tillfrågade har det kommit fram att enkäten var för lång och därför tror vi att resultatet kan ha påverkats. Däremot genererade enkäten en hel del förslag på funktioner som vi inte själva hade tänkt på, vilket lett till att vi kunnat utöka vår behovsanalys ytterligare. Därför tror vi också att resultatet blev som det blev och vi har insett att vi inte kan utgå från frågeformulärets resultat som grund vid vårt val av tjänst. Vi valde Hälsokoll dels därför att denna tjänst var en av de mest aktuella enligt vår uppdragsgivare, och dels eftersom det var något som gruppmedlemmarna gemensamt hade störst intresse av.

För att kunna gå vidare och utforma krav har vi upprättat personas utifrån de teorier som Sharp et al. (2007) redogör för (se bilaga 7). De utförda personas framtogs med varierade egenskaper och karaktärsdrag och utifrån dessa har scenarios upprättats (se bilaga 8). Utifrån framtagna personas, scenarios och enkätsvaren har kraven kunnat genereras.

Gruppen genererade många krav på vad tjänsten skulle kunna innefatta, både sådant som kan förverkligas med dagens teknik, men även krav som kan komma att utformas vid senare skede, då tekniken har utvecklats ytterligare. Kraven staplades sedan upp i ett dokument samtidigt som de sorterades efter vilken typ av krav de innefattar (se bilaga 15). I vår kravspecifikation innebär de funktionella kraven, precis som Sharp et al. (2007) säger, beskrivningar på vad systemet ska kunna utföra. Datakraven grundar sig på teorin Sharp et al. (2007) redogör för om vilken typ, storlek, omfång samt värdet av nödvändig data. Dock har inga giltighetskrav fastställts. Sharp et al. (2007) skriver om användbarhetskrav som bl.a. beskriver hur enkelt systemet är att lära, hur säkert det är, vilken användbarhet samt hur effektivt det är. De krav som finnes under våra användbarhetskrav innefattar den typen av krav som beskriver hur enkelt systemet är använda och känna igen sig samt hur effektivt det är. Användbarheten har delats upp ytterligare under två kategorier – tillförlitlighet och säkerhet – vilket har gjorts för att tillhandahålla en lättare översikt av kravspecifikationen. De krav vi har valt att placera under användargränssnittskrav innefattar sådant som har att göra med hur sidan kommer att se ut. Prestandakrav, som enligt Sommerville (2004) innefattar ett icke-funktionellt krav, har vi valt att placera de krav som har att göra med hur systemet kommer att fungera rent prestandamässigt.

Då vi hade tagit fram tillräckligt med krav bestämde vi ett möte med vår beställare på Media-IT. Detta möte gav upphov till en diskussion om vilka krav som kunde vara rimliga, dock ville beställaren att vi skulle se så mycket framåt som möjligt. Efter ytterligare diskussioner inom gruppen fattades ett beslut om att avgränsa kraven och de mest relevanta och troligast genomförbara valdes ut. Slutligen prioriterades kraven

efter en av RAD:s riktlinjer som Avison och Fitzgerald (2003) redogör för som kallas för MoSCoW-regeln. MoSCoW-regeln är av stor vikt då de försäkrar att de upprättade kraven kan motiveras och att de klassificeras. Kravspecifikationen i sin helhet återfinnes i bilaga 15.

3.2.7 Low-fidelityprototyp

3.2.7.1 Framtagning

Sharp et al. (2007) rekommenderar att använda low-fidelityprototyper för att generera idéer. Därför har gruppen valt att använda sig av low-fidelityprototyper med samma anledningar som ibid. nämner att det är enkelt att utforma, ger snabb feedback och är billigt att producera. Utifrån de framtagna kraven skissades tre olika prototyper för vår webbsida upp (se bilaga 9). Metoden som valdes var indexkort och genom att skissa upp prototyperna på papper har det gett möjligheter till öppna diskussioner där övergripande interaktion med prototyperna getts. De tre low-fidelityprototyperna var utformade med olika menyval och gränssnittslösningar. Då gruppen ansåg att det var begränsad tid kvar av projektet valdes det att inte rita prototyperna för avancerat. Syftet med att producera prototyperna var att kunna testa dem mot användare för att se hur de föll sig i deras tycke och smak. Därtill intervjuades testepersonerna vilket skulle komplettera low-fidelityprototyperna (se bilaga 10). Intervjuns avsikt var att få fram så mycket information som möjligt som kan vara användbart vid utformningen av de senare tänkta high-fidelityprototypen.

3.2.7.2 Test

Det första testet genomfördes utifrån de tre low-fidelityskisser som tagits fram (se bilaga 9) där tio testpersoner valdes ut att agera slutanvändare. Sharp et al. (2007) påpekar att testningen bör ske på fem till tolv personer och utifrån detta påstående tyckte vi att tio stycken testpersoner kändes lagom för vårt projekt och denna typen av test. Testpersonerna valdes ut med ett åldersintervall mellan 15 och 55 år, med anledning till att ha ett varierande åldersomfång. Testet gick ut på att finna den mest accepterade designlösningen utifrån de framtagna skisserna. Fem av de tillfrågade var bekanta sedan tidigare och de fem resterande personerna valdes slumpmässigt ut på ett gym som projektgruppen besökte. Då undersökningen genomfördes introducerades hela testet och dess syfte förklarades för testpersonerna. Sharp et al. (2007) framställer teori om öppen, ostrukturerad intervju vilket vi använde som tillvägagångssätt. Genom att använda öppna, ostrukturerade intervjuer har vi fått fram en hel del information och åsikter. Normalt sett används laboratiorium vid utförandet av dessa typ av tester men p.g.a de omständigheter vi befinner oss i finns inget sådant att tillgå. Skisserna introducerades och en rad frågor ställdes av intervjuaren. Avslutningsvis fick testpersonerna svara på ett par öppna frågor (se bilaga 10).

3.2.7.3 Utvärdering

Som Sharp et al. (2007) nämner beror tidpunkten av utvärderingen på vilken produkt eller tjänst som ska utvärderas. Vi valde att utvärdera vårt första test som utfördes på tre framtagna low-fidelityprototyper. Vid det första användbarhetstestet som gjordes på low-fidelityskisserna använde vi ramverket DECIDE, som Sharp et al. (2007) beskriver. Innan användartesterna startades bestämdes de övergripande målen med utvärderingen (Determine), vilka vi fastställde till att ta reda på testpersonernas åsikter om gränssnittet såsom menyns placering, färgval, etc. Explore och Choose, som är steg två och tre, innebär enligt Sharp et al. (2007) att sammanställa de frågor som ska besvaras samt bestämma vilket tillvägagångssätt som ska användas. Detta valde gruppen att genomföra genom att hålla öppna intervjuer med testanvändarna som i sin tur ledde till diskussioner (se bilaga 10). Identify betyder att testpersonerna ska väljas ut (Sharp et al., 2007). Vi frågade fem personer som är bekanta sedan tidigare, dessutom tog vi oss till ett gym där vi slumpmässigt valde ut de fem restrerande testpersonerna. Detta gjordes med anledning av att variera åldersintervallet på de tillfrågade ytterligare. Nästa steg, Decide, innebär enligt Sharp et al. (2007) att bestämma hur de etiska frågorna ska behandlas. I vårt fall var detta inget som behövdes ta hänsyn till då denna utvärdering inte var något som kunde vara stötande för de tillfrågade. Evaluate som förklaras av Sharp et al. (2007) innebär att den insamlade datan ska utvärderas, vilket skedde i form av att gruppmedlemmarna höll ett möte där diskussioner om användarnas åsikter fördes. Efter det att testpersonernas åsikter sammanställts ritades slutliga low-fidelityprototyper upp, se bilaga 11, och utveckling av databasen och high-fidelity prototypen startade.

3.2.8 Databashantering

Vid utformningen av databasen utgick gruppen från de krav som genererats tidigare och skapade entiteter och relationer dem emellan i form av ett ER-diagram. De var huvudsakligen de funktionella kraven som gav upphov i entiteter.

Vi upptäckte ganska snart att alla de idéer som har genererats skulle bli extremt tidskrävande att förverkliga och leda till att projektet hamnar utanför de utsatta tidsramarna. Därför beslutades det att ett ER-diagram som tar hänsyn till alla de idéer som gruppen fått skulle ritas upp förhand, på papper. Detta diagram skulle vara en önskvärd modell av hur databasen *skulle kunna se ut* om gruppen hade haft mer tid på att genomföra projektet. Vi beslutade därför att ta med alla de funktionerna som krävdes, enligt vår kravspecifikation, även om vi visste att den egentliga databasen inte skulle se ut som ER-modellen. Anledning till varför databasen har fått förändras gentemot den tilltänkta ER-modellen är många och kommer att redovisas löpande, allt eftersom ER-diagrammet beskrivs utförligt nedan. ER-diagrammet, som beskrivs i texten återfinnes i bilaga 12.

3.2.8.1 ER-modellen

Hjärtat av diagrammet är entiteten `user`. Den innehåller allt som har att göra med användaren, så som dennes inloggningsuppgifter, e-postadress och mobilnummer. Där förvaras även användarens längd (i form av ett tal som anger antal centimeter). Att längden sparas direkt i anknytning till användaren innebär att en användare endast kan ha en längd och när denna förändras och en uppdatering görs, kommer den gamla längden att försvinna. Detta är ett medvetet val, då vi gör ett antagande att en användare inte är lika intresserad av att hålla koll på sin längd som på sin vikt, då den förstnämnda inte förändras anmärkningsvärt hos en vuxen människa. Ett undantag där längden spelar en stor roll är barn och ungdomar, som fortfarande växer i snabb takt. Hälsokoll är dock inte menat att användas av ungdomar under 16 år och därför kan denna "brist" förbises. Det är viktigt att poängtera att om det skulle uppstå ett framtida behov av att hålla koll på längden på ett mer detaljerat sätt (exempelvis som på vikten, vilket beskrivs i följande stycke), kan detta enkelt uppnås genom att lyfta ut längd som en egen entitet.

En viktig del av Hälsokoll är att kunna hålla koll på användarens vikt över en lång tidsperiod, i syfte att kunna föra statistik, se mönster, anpassa träningen, med mera. Därför finns vikten som en egen entitet, `weight`, med tillhörande egenskaper:

- `userId` – vilken användare det är frågan om
- `dateRegistred` – tidpunkt som mätningen avser
- `userWeight` – antal kilogram som användaren väger vid den specifika tidpunkten

Ett av kraven för Hälsokoll är att användaren måste välja en av profilerna. I den första versionen av systemet, som levereras nu, har gruppen valt att avgränsa databasen och låta användaren välja mellan tre olika profiler: viktminskning, viktökning eller behålla vikten. För att göra det enkelt att i framtiden utöka antal profiler har dessa placerats i en egen entitet: `profile`, med tillhörande egenskaper som profilens ID, namn och en beskrivning. En användare måste ha en – och endast en – profil.

Profilen i sin tur bestämmer vilka övningar användaren ska utföra och vilken mat som användaren bör äta. Med andra är entiteterna `exercise` och `recipe` relaterade till någon av profilerna. Relationen mellan de två och `profile` är många-till-många, av den anledning att en viss profil innehåller flera olika maträtter (detta får ses som en självklarhet), men även att en och samma maträtt (t.ex. tortellini i olivolja med färsk basilika) kan tillhöra flera olika profiler (t.ex. viktökning och behåll vikt). En många-till-många relation innebär att en kopplingstabell förekommer (Padron-McCarthy & Risch, 2005). Kopplingen mellan recepten och profilen är enkel och innehåller endast de två nödvändiga ID:n. Däremot är kopplingen mellan profilen och träningsövningar något mer komplicerad, då den förutom ID:n innehåller två egenskaper: sets och reps (det vill säga antal set och repetitioner som övningen ska bestå av). Anledningen till att sets och reps placeras i kopplingen, istället för direkt i övningar är att en och samma övning kan behöva upprepas olika många gånger beroende på profil. Till exempel kan övningen "bänkpress" förekomma i alla tre profilerna som finns tillgängliga i nuläget, men i profilen "viktökning" ska övningen utföras enligt 3-6 (tre

set, sex repetitioner), i behåll vikt-profilen 4-10 och viktminskningsprofilen 3-15. För att hålla koll på hur tungt en viss användare lyfter under en viss övning vid en bestämd tidpunkt placeras denna data i en kopplingstabell mellan entiteterna `user` och `exercise`.

Ett av kraven är att användaren ska ha tillgång till en databas med artiklar. För att enkelt kunna avgränsa sökningen kommer varje artikel att kategoriseras, under ett visst ämne. Dessutom ska en användare kunna spara sina favoritartiklar. Artiklarna är en egen entitet, `articles`, där det förutom titeln och själva artikeltexten, sparas även när artikeln publicerades och av vilken typ den är. Det sistnämnda fältet innehåller en numerisk referens till någon av de typerna som finns i entiteten `articleType`. Detta gör det enkelt att med tiden fylla på med fler artikeltyper allt eftersom behovet uppstår, samtidigt som databasens storlek minimeras, då endast siffror lagras i `articles`, istället för textsträngar. Dessutom förekommer en koppling till, mellan `articles` och `user`, där ID:n för artikeln och användaren sparas för att tillåta en många-till-många relation. Den är tänkt att användas i syfte att låta användaren spara favoritartiklar.

Både träningsövningar och matlagningsrecept har en svårighetsgrad som är mycket flexibel då den bestäms helt och hållet i en egen entitet, `level`. Denna entitet kan innehålla exempelvis en betygssättning enligt med numeriska värden (t.ex. 1 - 5) eller språkliga konstruktioner (t.ex. nybörjare - avancerad). Då de egentliga värden sparas i entiteten `level`, medan `exercise` och `recipe` innehåller endast ID-nummer som refererar till en viss svårighetsgrad, undviks redundansen i databasen.

För att få kostdelen av systemet att fungera optimalt, kom gruppen fram till att det bästa – om än mest svårgenomförliga – lösningen är att lägga ut alla ingredienser som förekommer i alla de olika recepten i en egen entitet. Därigenom undviks redundansen och databasens storlek minimeras. Ett exempel är olika varianter av recept på lasagne där ”lasagneplattor” och ”nötkött” förekommer i varje recept, kan istället bytas ut mot tal, som ”64” och ”13” (d.v.s. ID-nummer för lasagneplattor respektive nötkött i `ingredient`-entiteten). Självfallet kommer mängden av en viss ingrediens att variera från recept till recept, och därför sparas denna relation i en kopplingstabell mellan `recipe` och `ingredient`.

En annan anledning till att gruppen anser att det är nödvändigt att lyfta ut `ingredient` är de olika användarnas matpreferenser. Systemet ger möjlighet för användaren att helt anpassa kostdelen, så att endast de relevanta (för användaren) recept visas. Till exempel, om användaren är vegetarian har hon/han ingen nytta av att få fram ett tiotal sätt att tillaga köttfärs på. En annan grupp av kostpreferenser har att göra med sjukdomar som laktosintolerans eller diabetes. Därför är det viktigt att de recepten som presenteras för en viss användare endast innehåller ingredienser som denne kan förtära. Lösningen på problemet är att skapa en entitet, som här fått namnet `specialFoodType`. Denna är tänkt att innehålla ett ID-nummer och ett namn, t.ex. ”1/diabetes”, ”2/laktos”, ”3/gluten”, med flera. Denna entitet är relaterad till `user` och `ingredient` genom två kopplingstabeller:

- Användaren kopplas ihop med en viss sort `specialFoodType` i en kopplingstabellen, som i ER-diagrammet betecknas med `can't eat`.

Anledningen till att det finns en kopplingstabell är relationen många-till-många (eftersom en användare kan vara t.ex. både glutenallergiker och vegetarian), och vid den sambandstypen förekommer kopplingstabeller (Padron-McCarthy & Risch, 2005).

- Den andra delen av kopplingen är där en viss ingrediens kopplas ihop med en viss typ av `specialFoodType` (som t.ex. kopplas `ingredient` ”23/vetemjöl” med `specialFoodType` ”3/laktos). Även här är relationen många-till-många, då en viss ingrediens kan höra till flera stycken `specialFoodType`, och viceversa.

Komplexiteten som uppstår i och med skapandet av `ingredient`-entiteten, är den största anledningen till att ER-diagrammet inte har kunnat implementeras. Uppbyggnaden av en databas med alla världens ingredienser, samt hur de olika sjukdomar/allergier och/eller religiösa/etiska övertygelse påverkar möjligheten att förtära en viss ingrediens av en användare, ligger långt utanför ramarna för projektet. Ändå vill gruppen redogöra för att även om prototypen inte innehåller denna funktionalitet, är steget inte långt till att implementera den, då den teoretiska grunden är – i och med ER-diagrammet – lagd.

3.2.8.2 Databasen

Databasen som byggdes för det här projektet har baserats på den ER-modellen som beskrivits utförligt i avsnittet ovan. Av de ovannämnda anledningar har vissa delar fått plockas ut och förenklats, för att göra projektet genomförbart under den utsatta tidsramen.

Databasen implementerades i Microsofts version av SQL, nämligen SQL Server 2005. Anledning till att vi valde denna databashanterare är dels att det är en relationsdatabashanterare (Padron-McCarthy &Risch, 2005) och dels den goda integrationen mellan SQL Server och den valda utvecklingsplattformen, nämligen Visual Studio 2005. Även om .NET har möjlighet att kopplas ihop med ett flertal andra databaser, genom s.k. ”providers”, har vi för det här projektet valt att använda den förvalda inställningar för databashantering i .NET, vilket på vår utvecklingsdator var en Microsoft SQL-databas inkapslad i en .mdf-fil.

Gruppen har valt att utnyttja de nya funktionerna i ASP.NET 2.0 fullt ut, och en av koncepterna som påverkar databasen är `Membership` (Evjen et al., 2006). Genom `Membership` tillhandahålls säkra metoder för registrering av användare, inloggning, återskapning av glömda lösenord, spärr av användare och förändring av lösenord. Det finns möjlighet att helt anpassa `Membership` genom att skapa egna regler för hur och var data ska lagras, men det är ett relativt stort område som gruppen inte har haft tid att fördjupa sig i. Därför används de förvalda inställningar, där `Membership` skapar en databas, ASPNETDB, och fyller den med de tabeller och data som behövs. Detta resulterar i att användarnamn, lösenord (i krypterad form) och användar-ID, med mera, sparas i en fristående databas, ASPNETDB.

Databasen som skapats för att lagra allt som behövs för Hälsokoll, exklusive inloggningsuppgifter, har döpts till Halsokoll.mdf. Databasen har byggts för att

uppfylla 3NF, som både Connolly och Begg (2002) samt Padron-McCarthy och Risch (2005) skriver om. Ett diagram över de tabeller som utgör databasen samt relationerna dem emellan kan ses i bilaga 13.

Som ett resultat av valet att använda `Membership`, innehåller gruppens databas Halsokoll.mdf, inga inloggningsuppgifter. Ändå har det beslutats att tabellen `user` bör finnas kvar (även om den nu är tom, eftersom inloggningsuppgifterna tillhandahålls av `Membership` och lagras i dess interna databas, ASPNETDB), för att underlätta en eventuell senare anpassning av `Membership`. Tabellen `user` utgör här, liksom i ER-modellen, en central del av databasen. Den kopplas med statistiska tabeller (som de som håller koll på användarens vikt och längd), inställningstabeller (ett exempel är `userFavoriteArticles`, som lagrar användarens favoritartiklar) och profiltabellen. En stor del av databasdesignen är lik den som beskrivits i stycket ovan, om ER-modellen, och därför kommer beskrivningen nedan att behandla dessa endast i ett kort stycke.

De tillgängliga profilerna (i nuläget tre stycken, samma som i ER-modellen) har lagts ut i en egen tabell för att underlätta en framtida utökning. Detta är ett av exemplen på hur systemet uppfyller skalbarhetskravet. En profil tillhandahåller, genom två kopplingstabeller, ett antal matlagningsrecept och träningsövningar. Recepten och övningar har en viss svårighetsgrad, och denna återfinnes i tabellen `level`. För att systemet ska veta vilka övningar som ingår i en viss profil finns tabellen `profileExcercise`. I denna anges även hur många repetitioner och set som ska göras för varje övning i en viss profil. Dessutom håller systemet koll på vilken övning en viss användare utförde, samt vid vilken tidpunkt och med vilken belastning det skedde. Data om det lagras i kopplingstabellen `executedExcercise`.

3.2.9 Design för mobila enheter

En mock-up är en modell av ett föremål som saknar funktioner och egenskaper menar Sharp et al. (2007). Vi har valt att ta fram mock-ups till mobila enheter för att förmedla en känsla över hur det kan komma att se ut så småningom, vilka illustreras i figurerna nedan. Sharp et al. (2007) beskriver brainstorming som en idégenereringsmetod. De framtagna mock-ups togs fram under en brainstormsprocess som gruppen höll, vilka återfinnes i bilaga 4.

Figur 5: *Log-in funktion*

Hjelm (2004) påpekar att länkar ska vara understrukna. Precis som figur 5 visar följer vi Hjelms (2004) teori då användaren ska logga in i systemet, länken logga in, har valts att vara understruken. Vi har utgått från att det ska vara så enkelt som möjligt att använda. Precis som Marsden (2006) skriver ligger fokus i gränssnittsdesign på känslan och det detaljerade utseendet vilket vi tagit hänsyn till vid utformningen av våra mock-ups.

Figur 6*: Meny med valmöjligheter*

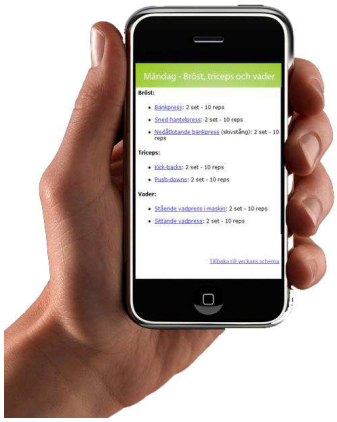
Figur 7*: Schema*

Då användaren väl är inloggad tillhandahålls en meny med de valmöjligheter som finns, se figur 6. Menyn är utformad i form av ikoner eftersom projektgruppen anser det vara estetiskt tilltagande och ger en förmedlan av framtidskänsla.

Då användaren gjort sitt val i menyn, exempelvis kalender, tillhandahålls ett schema som användaren själv kan anpassa via webbsidan och där bestämma om det t.ex. ska

visas efter dag, vecka eller månad (se figur 7). För att dessa funktioner ska fungera optimalt krävs det att användaren har registrerat en önskvärd profil via webbsidan.

Om användaren väljer att se hur dagens träningsschema ser ut visas övningarna detaljerat och en länkad text tillhandahålls, se figur 8. Därifrån kan användaren navigera sig vidare för att se träningsinstruktioner mer utförligt. Hjelm (2004) säger att användaren inte har mycket nytta av bilder och poängterar att då det visas bilder ska de ha ett samband med den text som erhålls. Det sistnämnda påståendet är just det vi har tagit hänsyn till på så sätt att bilder visas hur respektive övning ska utföras tillsammans med en textad beskrivning. Dessutom har gruppen haft i åtanke att träningsinstruktionerna så småningom ska kunna visas i videoformat så att användaren själv kan välja i vilken form instruktionerna ska ges, se figur 9. Hjelm (2004) föredrar att det används bilder med färre färger för att överföring ska kunna ske snabbare. Paelke et al. (2003) tillägger att endast ett begränsat antal färger bör visas på mobila enheter. Påståendena som Paelke et al. (2003) och Hjelm (2004) redovisar håller vi inte riktigt med om då vi anser att tekniken utvecklas allteftersom och de senaste åren har de mobila enheterna ständigt förbättrats och dessutom går det snabbare att överföra data. Därtill är mock-ups:en utformande med avsikt att de ska passa in i framtiden.



***Figur 8**: Träningsschema*

***Figur 9**: Träningsinstruktioner*

Något vidare utvecklingsarbete med de framtagna mock-ups till en fungerande prototyp har inte uträttats då projektet varit begränsat i tid och omfattning. Fokus har istället lagts på att utforma prototypen för webbsida och dokumentation.

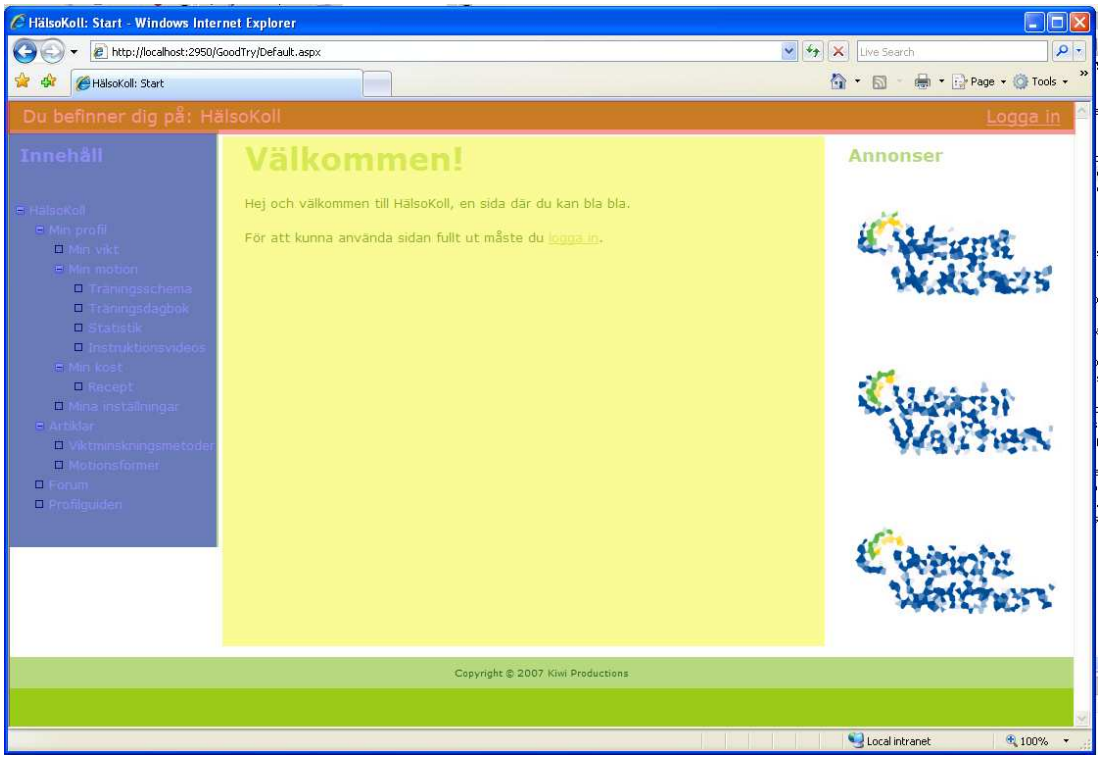
3.2.10 High-fidelityprototyp

3.2.10.1 Webbdesign

Webbplatsen skapades i Visual Studio 2005 och de mer specifika delarna som berör användning av tekniska finesser behandlas närmare under avsnittet ”Verktyg”.

Vid utformningen av webbplatsen tog vi hänsyn till Veens (2000) nyckelfrågor som en användare ska kunna besvara när denne är inne på sidan. Nedan förklaras de tre olika områden som författaren redogör för och tillämpas med vårt system i fokus (se figur 10):

1. Högst upp, markerat i orange, ser användaren var denne befinner sig på Hälsokoll och om denne är inloggad eller ej. Om en användare går in på en specifik meny, t.ex. *Min Träning*, ändras rubriken till *HälsoKoll > Min Träning*. Genom att erbjuda användaren denna funktion uppfyller vi Veens (2000) önskan om att ge svar på frågan: var är jag?
2. Till vänster, markerat i blått, finns en meny i form av en trädvy. Därifrån kan användaren navigera sig fram till olika sidor på webbplatsen vilket stämmer in på den teorin som Veen (2000) skriver att denna del bör tillhandahålla.
3. I mitten, markerat i gult, presenteras den viktiga informationen från de olika sidor som användaren navigerat sig till. Här besvaras frågan som användaren ställer: vad finns här? (Veen, 2000)



Figur 10: *High-fidelityprototypen med färgmarkerade områden som anknyter till de tre nyckelfrågor Veen (2000) framställer.*

Av det utformade testet som gjordes på low-fidelityskisserna ansåg åtta av tio testpersoner att menyn skulle vara vänsterjusterad. Detta stödjer vårt argument till den valda placeringen av menyn.

Nielsen [4] redogör för CSS som underlättar webbdesign, då anpassning av alla sidor till en specifik formatering görs. När det beslutades vilka färger, textformat, storlekar och marginaler vi skulle använda, togs CSS till hjälp. Färgvalet har tagits fram utifrån

testet som gjordes på low-fidelityskisser där majoriteten av testanvändarna ansåg att hälsa förknippas med gröna färger (se bilaga 10).

Nielsen beskriver att en webbplats ska vara enkel men ändå tydlig, med innehåll som är kort, exakt och problemfritt att skumma igenom [4]. Vår önskan var att utforma en webbplats som överrensstämde med föregående påstående. Därutöver nämner Sharp et al. (2007) att flera interaktionsdesigners anser att webbplatsen ska vara estetiskt behaglig vilket vi även tog till oss då vi utformade prototypen. Användaren utsätts inte för allt för många bilder som tar på överföringstiden eller anses som ett störmoment. De bilder som en användare kan finna som ett störmoment, är annonserna som finns högerjusterat på sidan. Anledningen till att vi tog med dem, trots en fullständig medvetenhet om att det inte är populärt fanns, var att vår beställare hade kunnat ge sponsorerna den platsen om det skulle behövas. Annonsutrymmet är för övrigt väldigt enkelt att ta bort och sidan kan då bli mer behaglig för användaren. Sidan är inte tänkt att vara helt utan illustrationer, snarare tvärtom, det är nämligen tänkt att det ska återfinnas instruktionsvideos och bilder för hur olika övningar ska genomföras på rätt sätt. Genom att både tillhandahålla instruktionsvideos och bilder kan användaren själv välja vilken typ av format som önskas ses, genom att länka i form av miniatyrer eller text.

Precis som Hjelm (2004) påpekar – att en webbplats inte ska fråga användaren efter uppgifter som inte kommer att användas – gör inte vi det på webbplatsen. Anledningen till detta är att vi inte vill ha en databas full med obehövlig data som inte används till något. Den specifika informationen vi efterfrågar är det krav på att uppge då en person önskar bli medlem. De uppgifter som behöver anges är:

- Användarnamn – ett unikt namn som användaren anger när denne ska logga in på sidan.
- Lösenord – ett lösenord som användaren anger i rutan för lösenord där texten maskeras och blir därmed dold, vilket görs för att användaren ska kunna hålla lösenordet privat.
- E-post – då användaren blivit medlem krävs det att en e-postadress registreras. Denna används för att skicka en bekräftelse till användaren, dessutom kan information om webbplatsen och dess regler sändas. E-postadressen används även om användaren råkar glömma sitt lösenord. Av denna anledning måste e-postadressen vara unik.

Då användaren registrerats som medlem och vill skapa en användarprofil bör följande uppgifter registreras:

- Förnamn och efternamn – används för att webbplatsen ska kunna vända sig mer personligt till användaren.
- Födelsedatum – kan användas för att specificera tränings- och kostprofiler utifrån användarens ålder.
- Mobiltelefonnummer – används för att användaren ska kunna få information till sin mobil.

De funktioner som återfinnes på webbplatsen har kommit fram utifrån de framtagna kraven samt de användartester som utformades på tre low-fidelityskisser. Ur testerna framkom det att det ansågs som önskvärt att det skulle finnas artiklar som berör hälsa.

Dessutom påpekade fem av de tio tillfrågade att de önskade ett forum, där de kan dela tips och idéer med andra användare. Genom detta forum kan dessutom de som kämpar mot samma mål stötta och uppmuntra varandra.

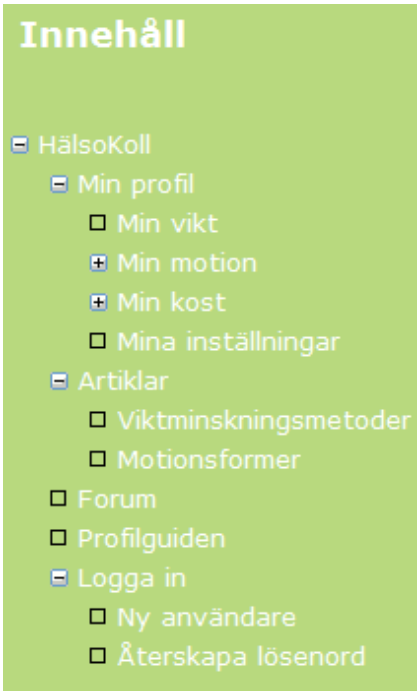
3.2.10.2 Verktyg

För att implementera webbplatsen användes Microsofts lösning för skapande av webbapplikationer: ASP.NET. Anledning till detta är ett av kraven, som direkt säger att tjänsten ska implementeras i ASP.NET.

Gruppen såg det som en självklarhet att använda de senaste tillgängliga verktygen på området, i syfte att skapa så behaglig användarupplevelse som möjligt. ASP.NET i version 2.0 tillhandahåller ett antal tekniker och lösningar som, enligt Evjen et al. (2006), Darie och Watson (2006), ger utvecklarna möjlighet till att effektivisera sitt arbete och fokusera på det väsentliga för användaren.

Inledningsvis valde gruppen att skapa en s.k. *master page* som skulle vara en mall för alla de sidor som ingår i webbplatsen. Denna mall utformades efter det designförslaget som var mest populärt under användartestet av low-fidelityprototyper. För att få den grafiska layouten enhetlig applicerades en `style sheet` på mallen. Därefter användes mallen för att skapa de olika sidorna som skulle utgöra webbplatsen. Anledning till att vi valde att arbeta med `master pages` är den fördelen som nämns av Darie och Watson (2006), nämligen att utvecklaren kan vara säker på att ett konsistent utseende förekommer på alla webbsidor som använder mallen. Vidare ville gruppen framtidssäkra prototypen, genom att göra det enkelt att genomföra stora förändringar till utseendet på webbplatsen, om det – efter testet av high-fidelityprototypen – skulle visa sig vara nödvändigt.

För att ytterligare försäkras om att prototypen skulle vara lätt att modifiera efter framtida behov har ett navigationssystem som är enkelt att förändra implementerats. Evjen et al. (2006) skriver att ASP.NET:s lösning är att skapa en s.k. `site map` och därefter använda en kontroll som skapar menystruktur utifrån `site map`-filen. Gruppen valde att använda sig av en trädvylikande kontroll, där huvudsidorna har olika undersidor och på sätt uppstår det en visuell menystruktur (se figur 11). De huvudgrupper som finns med i trädvyn motsvarar till stor del de huvudgrupper av funktionella krav som förekommer i kravspecifikationen. En annan anledning till att trädvyn används, är att gruppen ansåg att det är ett bra sätt att ha relevant innehåll i det blåmarkerade område i figur 4 (som presenterats i avsnittet Webbdesign), vars syfte är att svara på Veens (2000) andra fråga som användaren bör kunna besvara när denne besöker en webbsida: ”Vart kan jag gå?”. Ett argument som stärker gruppens val av trädvy som navigationselement är att alla de testanvändare som kört prototypen anser att navigationens utformning och placering är utmärkt.



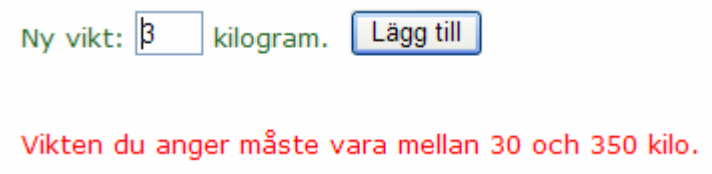
Figur 11: *Kontrollen TreeView som tillhandahåller en strukturerad meny.*

Ännu en anledning till att använda `site map` är att ASP.NET tillhandahåller en kontroll som väl lämpas att placeras i det orangefärgade område i Figur 4. Detta område ska enligt Veen (2000) ge svar på användarens fråga ”Var är jag?”. Kontrollen, som heter `SiteMapPath` (se figur 12) visar tydligt var användaren befinner sig just nu, och hur den nuvarande sidan förhåller sig till andra i menyhierarkin.

Figur 12: *Kontrollen* `SiteMapPath`

En av de mest vitala delar av systemet är hantering av användarkonton, inklusive registrering, inloggning, sessionshantering, blockering av användare, återskapande av lösenord och borttagning av användarkonton. För att systemet skulle tillhandahålla denna funktionalitet valde gruppen att arbeta med ASP.NET:s ramverk `Membership and role service`. En anledning till att detta ramverk valdes är att det underlättar framtagning av många av de ovannämnda funktionerna och minimerar utvecklingstiden. Eftersom gruppen hade relativt ont om tid att lägga på själva utvecklingen (då tjänsten är så omfattande), ansågs alla hjälpmedel som förkortade utvecklingstiden vara viktiga att utnyttjas. Därför var valet att använda ramverket enkelt eftersom, som Evjen et al. (2006) uttrycker det, förkortar `Membership` utvecklingstiden genom att tillhandahålla lösning för de vanligt förekommande problem som har med hantering av användare att göra. En annan anledning till att `Membership` och de kontroller som tillhandahålls av ramverket användes, var att de komponenter som ingår är utvecklade av Microsoft och har testats väl innan de kom att levereras med .NET. Därför tror vi, liksom Evjen et al. (2006) skriver, att komponenterna har troligtvis få buggar och säkerheten kan anses vara hög.

Ett av de senaste tillskotten till webbutveckling heter AJAX och möjliggör skapande av webbgränssnitt som mer efterliknar traditionella grafiska användargränssnitt. Ett exempel där tekniken utnyttjas är validering av inmatningsfält, där användare får en varning om fältet innehåller ett otillåtet värde, redan innan sidan skickats till servern (se Figur 13).



Figur 13: *Exempel på hur AJAX används för att validera inmatningsfält.*

Ett annat exempel där AJAX används är en uppdateringsindikator, som låter användaren se att systemet arbetar (se Figur 14). Anledning till att gruppen valde att arbeta med AJAX är att vi ansåg det vara positivt att, i så stor utsträckning som möjligt, få tjänsten att likna en applikation.

Figur 14: *Exempel på hur AJAX används för att förbättra systemet interaktion med användaren.*

3.2.10.3 Test

Trots att Hjelm (2004) skriver att det är svårt att hitta en testmetod för webben som ger en rättvisande indikation, ville gruppen ändå utföra tester på high-fidelityprototypen, eftersom de anses viktiga av Sharp, et al. (2007). Test ska, enligt Sharp et al. (2007), undersöka huruvida produkten är användbar för det syfte som den är utformad för, och hur effektiv den är. Gruppen valde att förbereda ett sådant test och genomföra det vid ett senare skede, när det fanns en fungerande high-fidelityprototyp att testa på. Det valet stödjer vi återigen på teorin från Sharp et al. (2007), där författarna anser att test bör genomföras i ett senare skede av utvecklingsfasen.

Sharp et. al (2007) skriver att fem till tolv individer är ett rimligt antal testanvändare, men det går bra att använda sig av färre, om budgeten eller tiden inte räcker till. Gruppen ansåg att minikravet var lämpligt, då tiden som fanns för att genomföra testet var knapp, och valde därför att genomföra testet på fem personer i åldern 20-34 år. De fem var studenter på IT-ekonomprogrammet, och ansågs kunna ge bra omdöme gällande tjänstens utformning, funktionalitet och marknadspotential.

Sharp et. al (2007) menar att testing bör utföras i speciellt konstruerade laboratorium som är anpassade till det specifika testet. Då gruppen inte har tillgång till ett laboratorium, på grund av de höga kostnader som skulle uppstå vid uppbyggnaden av ett sådant, utfördes testet i en av Halmstad Högskolas datorsalar. Prototypen kördes

på en PC-dator med de två största webbläsare på marknaden, Mozilla Firefox och Microsoft Internet Explorer. Testpersonen satt direkt framför datorn med standardutrustning (i form av skärm, tangentbord och mus) medan två observatörer iakttog händelseförloppet. En av observatörerna introducerade prototypen och ställde specifika frågor kring användbarheten (dessa återfinnes i bilaga 14). Den andra observatören antecknade reaktionerna som väcktes hos användaren när denne talade om vilka tankar som uppstår under användning av tjänsten, något som Hjelm (2004) benämner ”att tänkta högt”. Den öppna och ostrukturerade intervjuformen användes, med motivationen att denna, enligt Sharp et. al (2007), är lämplig att användas för att undersöka åsikter och generera mycket information.

Innan testpersonerna fick prova prototypen förklarades tjänstens bakgrund, vad prototypen är tänkt att utföra och tillståndet med prototypen de testar (d.v.s. att prototypen inte är färdig och därför inte fullt funktionell). Därefter påbörjades testet och testpersonen fick navigera fritt på webbplatsen. Det tog inte lång tid innan användarna insåg att det inte går att komma långt utan att vara en inloggad, vilket ledde till att alla testpersoner navigerade till registreringen för nya användare. Att skapa ett användarkonto var enligt samtliga testpersoner problemfritt. Två testpersoner valde att felsöka inloggningen genom att skriva in fel lösenord och se hur systemet skulle reagera. Resultatet blev att valideringen bad användaren att kontrollera sina inloggningsuppgifter, då de var felaktiga. Utvecklingsteamet tolkade det som att inloggningen fungerar korrekt och kan anses vara stabil.

När testanvändaren loggade in bemöttes denne av en text som hänvisade till profilguiden, som fungerar som hjälp att få fram en personlig profil. Guiden, som i prototypen ännu inte var fullt utvecklad, bestod av tre enkla frågor som testpersonerna inte hade några problem med att besvara. Samtliga testpersoner fick, efter att ha genomfört guiden, veta att den profil som guiden valde som lämpligast åt dem var förprogrammerad (vilket innebär att alla deltagare fick samma profil som förval). Anledningen till att gruppen valde att förprogrammera en inställning som standard var att det skulle krävas betydligt mer tid, för forskning kring frågorna och utveckling av en guide som skulle ge rättvisande svar, än vad gruppen hade till sitt förfogande.

Efter att användarna skapat en profil kom de till sidan ”Min profil” där de fick testa sig fram till olika funktioner. Samtliga användare hade stora synpunkter gällande utformning av menyvalen på sidan med inställningar. Dessa synpunkter låg till grund för en ny version av inställningssidan.

Gällande färgvalet tyckte samtliga fem testpersoner att det lämpades väl för en sida som Hälsokoll, och kommentarerna var bl.a.: ”Passande färg, jag tänker på ordet nyttigt”, ”Denna färg associerar jag till hälsa”, ”Snyggt!” och ”Fina hälsofärger”.

3.2.10.4 Utvärdering

Den andra utvärderingen gjordes i ett senare skede, på high-fidelityprototypen. Determine innebär enligt Sharp et al. (2007) att bestämma målen för utvärderingen, som bestämdes till att undersöka hur väl systemet fungerar, om det är enkelt att förstå, reaktionen på funktioner, färgval och få så mycket feedback från testpersonerna som

möjligt. Steget Identify innebär att testpersonerna identifieras (Sharp et al., 2007). Vi identifierade fem stycken studiekamrater, dessa kändes lämpliga då de har egna erfarenheter av prototyputveckling och ses därför tillhandahålla ett gott omdöme. Därefter bestämdes tillvägagångsättet för att utföra utvärderingen, vilket vi bestämde skulle ske utifrån observationer och öppna intervjufrågor. Hjelm (2004) förespråkar att det kan vara givande att låta testpersonerna tänka högt då de interagerar med prototypen, vilket vi bad testpersonerna att göra. Samtidigt noterade utvärderarna testpersonernas reaktioner och noterade dessa. Intervjufrågorna skrevs ner och återfinnes i bilaga 14. Vid denna utvärdering var det, precis som vid första utvärderingen, ingenting som skulle kunna vara stötande för testpersonerna och därför behövde vi inte ta hänsyn till de etiska frågorna. Slutligen anordnade projektgruppen ett möte där testpersonernas reaktioner och åsikter diskuterades och vidare utveckling av prototypen kunde göras utifrån testanvändarnas åsikter i fokus.

3.2.11 Prissättning

Kotler et al. (2002) redogör för tre olika prissättningssmetoder; kostnadsbaserad, värdebaserad och konkurrensbaserad. Utifrån dessa metoder anser vi att Hälsokollstjänsten borde sätta sitt pris efter både värdebaserad och konkurrentbaserad prissättningsmetod. Då Hälsokoll skapar ett värde för kunden i form av en personlig guide där kundens målsättning sätts i fokus, därtill uppnår kunden en bättre hälsa vilket ofta kan betraktas som ovärderligt. Därför kan också prissättningen bestämmas därefter. Givetvis måste hänsyn tas till konkurrenternas prissättning, sätts det ett för högt pris väljs konkurrenterna före oss även om vi är unika med vår tjänst. Då vi har genomfört test på utvalda personer har en intressant aspekt varit priset som användarna är villiga att betala. Av våra fem testpersoner uppgavs det ett relevant pris som låg mellan 39 och 300 kronor per månad och någon ansåg att “man får en personlig tränare billigt”. Dessutom ansågs det som önskvärt att tjänsten skulle erbjuda studentrabatt.

4 Diskussion

Prototypen som framställdes under projektets gång är på intet sätt komplett. Detta var planerat, och kan utläsas tydligt i vårt syfte.

Projektet startade med att gruppen utfomrade en projektplanering bl.a. över projektets syfte, vilka uppgifter var och en ansvarade för samt en tidsplan. Denna har gruppen försökt att arbeta efter men kom ganska snabbt till konsensus om svårigheterna med att uppskatta tid och ordning på genomförandet då ingen av oss har någon större erfarenhet från projektarbete i denna omfattning sedan tidigare. Vi vill poängtera att tidsfristen för projektet har varit knapp och en ständig tidspress har funnits. Vi hade sett det som önskvärt att de utsatta tidsramarna för projektet hade varit bredare då vi anser att prototypen i så fall hade kunnat bli mer funktionellt utbyggd.

Gällande riskerna som indentiferats under avsnittet Riskhantering kan gruppen konstatera att många har identiferats korrekt. Som de två största riskerna kan vi se bristande kunskaper inom området samt felbedömning av tidsplanen. Dessa två hänger ihop på så sätt att för att kunskapen ska kunna inhämtas måste tiden tas från andra sidoprojekt inom arbetet, samtidigt som, om inte tillräckligt med tid läggs på kunskapsinhämtningen, kommer inte arbetet att kunna genomföras. En annan risk som uppenbarades var bristande rättigheter gällande serveråtkomst. Detta har lett till att prototypen ej kunnat laddas upp till en publik server och testas på fler personer eller presenterats under delredovisningar. Ännu en risk som kom att förverkligas var felprioritering. Gruppen inledde projektet med att studera litteratur och börja skriva teorier som ansågs relevanta för projektet. I efterhand kan det dock konstateras att det lades ned för mycket tid på att skriva teoridelen.

XP som utvecklingsmetod var svår att tillämpa fullt ut i vårt projekt. På grund av arbetets omfattning i relation till den tillgängliga tiden för genomförande av projektet, har gruppen fått förändra sättet som prototypen utvecklats på. Den främsta anledningen till att vi valde modellen var att den anses vara flexibel vid nya, komplexa projekt. Vi anpassade modellen, genom att ta in delar från andra utvecklingsmodeller och ta bort några från XP. Från början hade vi tänkt oss att parprogrammering skulle tillämpas under programmeringsfasen, men i efterhand visade det sig vara ett alldeles för tidskrävande tillvägagångssätt.

Något som projektgruppen ansåg vara svårt under projektets gång var att utföra testerna på ett lämpligt sätt. Helst hade vi velat testa den slutliga prototypen i ett laboratorium, som normalt görs, men den knappa tiden och de höga kostnaderna gav oss ingen möjlighet till detta. Istället utfördes två olika tester varav det första var på de tre framtagna low-fidelityskisserna. Det andra testet ufördes på 5 personer som fick prova vår high-fidelityprototyp samtidigt som två av gruppmedlemmarna observerade och ställde frågor. Detta var mycket intressant och gruppmedlemmarna kände att det gav oss en hel del lärdom över prototypens starka och svaga sidor.

Uppgiften som vi tog på oss kändes väldigt komplex då vi fick relativt fria händer från vår beställare. Gruppmedlemmarna upplevde dessutom att det var svårt att kunna specificera de krav som var mest tillämpbara då vi gärna hade fortsatt prototyputvecklingen ytterligare. På grund av den begränsade tiden fick en variant av prototyper byggas, som enligt Sharp et al. (2007) benämns för vertikal prototyp. Denna uppfyller de kraven som är avbockade i vår kravspecifikation (se bilaga 15). Vi anser att prioritering av kraven, som skett efter det som Avison och Fitzgerald (2003) kallar för MoSCoW-regeln, har fungerat bra och varit ett stort hjälpmedel vid avgränsning av funktionaliteten. Kravspecifikationen är relativt välutvecklad och innehåller många punkter som är tänkta för en framtida version av tjänsten och har därför ej implementerats i prototypen. Detsamma gäller kravspecifikation för mobila versionen av tjänsten (se bilaga 16).

Tack vare de regelbundna träffarna och ett gott samarbete med handledaren har feedback fåtts och förbättringar av dokumentationen kunnat göras. Dessutom har handledaren hjälpt oss med att finna lämplig litteratur. Beställaren har funnits tillgänglig för samarbete under projektets gång och vi har hållt ett antal möten som varit väldigt givande och inspirerande. Under möten med beställaren tycker vi, att det tydligt framkommit att resultatet av projektet bör vara väldigt framtidsinriktat och, i

vissa fall, ta tillvara på funktioner utrustning som ännu inte finns. Vi har försökt arbeta så långt som möjligt efter denna vision. Detta återspeglas bl.a. i att vår konceputella datamodell (ER-modellen, se bilaga 12) är utformad på ett sådant sätt som stödjer lagring av data som ännu inte används i den nuvarande versionen av prototypen. Även i kravspecifikationen kan en del framtidsinriktade funktioner utläsas. Dessutom har vi tagit fram, och prestentarat, mock-ups för en framtida version av tjänsten för mobila enheter. Däremot måste vi konstatera att den high-fidelityprototypen som levereras med projektet tyvärr saknar stöd för många av de tankar och idéer som genererats under tidigare faser. Som främsta anledning till detta, ser vi – förutom tidsbristen och avsaknad av fysiska enheter med nödvändig funktionalitet (som t.ex. bluetoothvågen) – kunskapsbrist.

Referenser

Andersen, S. E., Grude, V. K. & Haug, T. (1994). *Målinriktad projektstyrning.* Lund: Studentlitteratur

Apelkrans, M. & Åbom, C. (2001). *OOS/UML*. Studentlitteratur.

Avison, D. & Fitzgerald, G. (2003). *Information Systems Development - Methodologies, Techniques and Tools.* (3rd ed.). McGraw-Hill

Backman, J. (1998) *Rapporter och uppsatser*. Lund: Studentlitteratur

Bell, D. (2005). *Software Engineering for Students – A Programming Approach.* (4th ed.). Harlow: Pearson Education Limited

Boehm, B. W. (1988). A Spiral Model of Software Development and Enhancement. *IEEE Computer* 21(5), 61-72.

Carlson, H. & Nilsson, A. (2002). *Arbeta i projek*t!. Höganäs: Kommunlitteratur AB

Connolly, T. & Begg, C. (2002). Database Systems – A Practical Approach to Design, implementation, and Management. (3rd ed.). Harlow: Pearson Education Limited

Darie, C. & Watson, K. (2006). *Beginning ASP.NET 2.0 E-Commerce in C# 2005 – From Novice to Professional*. Berkeley: Apress

Eklund, S. (2002). *Arbeta i projekt – en introduktion*. Lund: Studentlitteratur

Eriksson, U. (2007). *Kravhantering för IT-system*. Lund: Studentlitteratur

Evjen, B., Hanselman, S., Muhammad, F., Sivakumar, S. & Rader, D. (2006). *Professional ASP.NET 2.0*. Indianapolis: Wiley Publishing

Hjelm, J. (2004). *Mobilisera webben*. Lund: Studentlitteratur

Jones, M. & Marsden, G. (2006). *Mobile Interaction Design*. West Sussex: Wiley.

Kotler, P., Armstrong, G., Saunders, J., & Wong, V. (2002). *Principles of Marketing (3rd European ed.)*. England: Pearson Education Limited

Marttala, A. & Karlsson, Å. (1999) *Projektboken – metod och styrning för lyckade projekt*. Lund: Studentlitteratur.

McBreen, P. (2003). *Questioning Extreme Programming*. Boston: Pearson Education, Inc.

McManus, J. & Wood-Harper, T. (2003). *Information systems project management methods, tools and techniques*. Boston: Pearson Higher Education

Padron-McCarthy, T. & Risch. T. (2005). *Databasteknik.* Lund: Studentlitteratur.

Paelke, V., Reimann, C. & Rosenbach. (2003). A Visualization Design Repository for Mobile Devices. [Elektronisk version**].** *Computer graphics, virtual reality, visualisation and interaction in Africa*, 57-62.

Sharp, H., Preece, J. & Rogers, Y. (2007). *Interaction Design - Beyond Human-Computer Interaction.* (2nd ed.). Wiley.

Sommerville, I. (2004). *Software Engineering*. (7th ed.). Addison-Wesley.

Veen, J. (2000). *The Art and Science of Web Design*. New Riders.

[1] [WWW-dokument] Wikipedia
http://upload.wikimedia.org/wikipedia/en/thumb/5/51/Waterfall_model.png/350px-Waterfall_model.png

[2] [WWW-dokument] Elanman
http://www.elanman.org/teaching/gmu/swe620-infs622/Graphics/spiral_model.gif

[3] [WWW-dokument] Extremeprogramming
http://www.extremeprogramming.org/

[4] [WWW-dokument] Useit.com Jakob Nielsen´s Website
www.useit.com

[5] [WWW-dokument] Wikipedia
http://en.wikipedia.org/wiki/Gantt_chart

[6] [WWW-dokument] Manifesto for Agile Software Development
http://www.agilemanifesto.org

[7] [WWW-dokument]
http://en.wikipedia.org/wiki/AJAX

[8] [WWW-dokument]
http://ajax.asp.net/docs/overview/default.aspx