Utvärdering, design och implementering av användargränssnitt för modellbaserat testverktyg

Jerry Malm

Master's Thesis

Department of Design Sciences
Lund University
ISRN:LUTMDN/TMAT-5129-SE

EAT 2009

Utvärdering, design och implementering av användargränssnitt för modellbaserat testverktyg

Evaluation, design and implementation of user interface for model based test tool

Författare: Jerry Malm

Handledare: Joakim Eriksson

Extern handledare: Bobby Kociski

Datum: 2009-07-13

LUNDS UNIVERSITET

Lunds Tekniska Högskola

Institutionen för Designvetenskaper

Abstract

The company System Verification was developing a model based test tool during the writing of this thesis. The test tool will be used to test software systems and for this an intuitive user interface is wanted. The purpose of this study was to examine how a user interface for a model based tool should look. Other things to examine were if there was any standard for such a user interface and what made current test and model tools good or bad. The first thing that was made was an evaluation of current test and model tools. The conclusions from this evaluation were used to design two user interfaces with different design principles. Two prototypes were implemented to test which design principles that should be kept and what things that needed to be changed. The user interface was then developed based on the results from these tests. To confirm that the new user interface was better than the previous ones a second round of tests were executed. Just like in the previous test phase a couple of advices about how the user interface could be improved were found and these got implemented during the third and final implementation phase. A couple of guide lines on how a model based test tool should look was given after the testing and implementation of the final prototype. These guide lines were based on the results from the tests. The results from the testing of the prototypes showed that the test persons felt that things like plainness, feedback, sorting and the possibility to work effectively were important for an interface of a model based test tool.

Key words

Model based testing, user interface, intuitive, evaluation, testing, guide lines

Sammanfattning

Företaget System Verification utvecklade vid denna rapports skrivande ett modellbaserat testverktyg. Testverktyget kommer att användas till att testa mjukvarusystem och till detta önskades ett intuitivt användargränssnitt. Syftet med denna studie var att undersöka hur ett användargränssnitt till ett modellbaserat testverktyg skulle kunna se ut. Andra saker som skulle undersökas var om det fanns någon standard för ett sådant gränssnitt och vad det är som gör befintliga test- och modelleringsverktyg bra eller dåliga. Det första som gjordes var en utvärdering av befintliga test- och modelleringsprogram. Slutsatserna från utvärderingen låg sedan som grund för hur två användargränssnitt med olika designprinciper skulle kunna se ut. Två prototyper implementerades för att testa vilka av dessa designprinciper som skulle behållas samt vilka saker som behövde ändras. Användargränssnittet utvecklades sedan utifrån resultaten av testerna. För att kontrollera att det nya användargränssnittet hade förbättrats mot tidigare versioner genomfördes en andra omgång testning. Precis som under föregående testomgång kunde ett antal råd för hur gränssnittet skulle kunna förbättras tas fram och dessa implementerades under den tredje och sista implementeringsfasen. Efter att den slutgiltiga prototypen hade testats och implementerats färdigt kunde ett antal riktlinjer för hur ett modellbaserat testverktyg skulle kunna se ut ges. Dessa riktlinjer baserades på de resultat som gavs från testerna. Resultaten ifrån testningarna av prototyperna visade att försökspersonerna upplevde att saker som tydlighet, återkoppling, sortering och möjlighet att effektivisera arbetet som viktiga för ett användargränssnitt till ett modellbaserat testverktyg.

Nyckelord

Modellbaserad testning, användargränssnitt, intuitivt, utvärdering, testning, riktlinjer

Förord

Det här examensarbetet är skrivet på Lunds Tekniska Högskola (LTH) [1] i samarbete med företaget System Verification från Malmö. Tack alla ni som hjälpt mig under arbetets gång, utan er hade det inte gått. Personer som jag skulle vilja tacka lite extra är mina handledare Joakim Eriksson från LTH och Bobby Kociski från System Verification som gett mig många värdefulla kommentarer och hjälpt mig med praktiska detaljer. Andra personer som bör nämnas lite extra är min flickvän Filippa Ekbladh, som testat tidiga versioner av prototyperna samt granskat denna rapport, och Jonas Gyllenspetz från System Verification, som tog sig tid att hitta detta arbete åt mig.

Jerry Malm

Student at Computer Science and Engineering

2009-07-13 Lund

Innehållsförteckning

1 Introduktion

Företaget System Verification [2] är ett företag från Malmö som har testning och testmanagement som fokus. Vid denna rapports skrivande, utvecklade System Verification ett verktyg för modellbaserad testning. Till detta verktyg önskades ett intuitivt användargränssnitt som skulle underlätta målgruppens arbetsprocess. I programmet skulle det gå att skapa modeller, generera testfall utifrån en modell samt exekvera testfall. Målgruppen var testdesigners och testexekverare som hade stor erfarenhet av datorer och testning sedan tidigare. Rapportens författare anlitades som examensarbetare för att ta reda på hur ett sådant användargränssnitt skulle kunna se ut för att i bästa möjliga mån tillgodose användarnas behov.

I dagens samhälle handlar allting mer och mer om utseende och häftiga funktioner. Tyvärr blir det i en del fall användarvänligheten som blir lidande till förmån för de båda ovanstående attributen. Det var därför positivt att höra att System Verification ville ha ett användargränssnitt som skulle fokusera på användarvänligheten. Varken ett snyggt yttre eller häftiga funktioner var något som skulle prioriteras. I många program som utvecklas för privat bruk glöms eller bortprioriteras, enligt författarens egna erfarenheter, ofta användbarheten. Detta för att det tar tid att designa och utveckla ett användargränssnitt som är användarvänligt. Ibland finns det inte tid och resurser nog för att skapa ett intuitivt användargränssnitt och ibland finns det helt enkelt inget behov för det då det enbart är utvecklarna själva som ska använda programmet. Självfallet kommer utvecklarna att tycka att användargränssnittet är lätt att använda då det är de själva som gjort det. De vet exakt hur programmet ska användas för att få ut bästa resultatet samt vilka fallgroparna är. Skulle däremot en användare som ej deltagit i utvecklingsprocessen sätta sig med programmet skulle hon antagligen inte tycka att användargränssnittet var särskilt intuitivt. Dessutom skulle användaren ej ha kunskap om fallgroparna som i värsta fall skulle leda till att programmet kraschade. Detta är ett problem som kan minska produktiviteten hos ett arbetslag om endast en minoritet av användarna deltagit i utvecklingen av programmet. Det är alltså viktigt att gränssnittet blir så pass intuitivt att personer som ej deltagit i utveckling av programmet tidigare också kan använda det på ett effektivt sätt utan att behöva gå en kurs. Samtidigt är det viktigt att effektiviteten hos programmet inte blir lidande på grund av att det ska göras användarvänligt för nybörjare.

Syftet med studien var att ta reda på hur ett gränssnitt för ett modellbaserat testverktyg skulle kunna se ut som gör det intuitivt att använda, båda för nya och erfarna användare. Andra mindre centrala frågeställningar som också kommer att behandlas i denna rapport är vad det är som gör befintliga testverktygs användargränssnitt bra eller dåliga samt om det finns någon standard för användargränssnitt för modellbaserade testverktyg.

Rapporten kan delas upp i två större delar. Först kommer en presentation av de olika arbetsmetoderna i kronologisk ordning och efter varje metodikavsnitt presenteras resultatet av arbetsmomentet. I den avslutande delen av rapporten presenteras slutsatser utifrån resultatet.

2 Genomförande

Metoderna som användes under examensarbetets gång beskrivs i kronologisk ordning så att arbetets gång lätt kan följas av läsaren. Efter varje metod presenteras resultatet ifrån föregående arbetsmoment för att läsaren ska slippa att bläddra för att ta reda på resultatet.

2.1 Litteraturstudie

Det första som gjordes var en litteraturstudie inom berörda områden. Med hjälp av handledarna Joakim Eriksson och Bobby Kociski identifierades ett antal lämpliga källor inom områdena användbarhet och modellbaserad testning. Dessa källor, som finns i referenslistan, studerades sedan för att hitta lämpliga metoder och utvärderingskriterier för att säkerställa arbetets kvalitet samt för att få en klar bild av modellbaserad testning. Utöver detta studerades även UML, Unified Modeling Language, eftersom det är en viktig del av modellbaserad testning.

2.1.1 Modellbaserad testning

Modellbaserad testning [3] är att utifrån en modell, som beskriver ett systems funktionalitet, skapa testfall. Det första som görs är att en modell specificeras för att efterlikna ett system. Detta kan antingen göras genom att skriva kod som beskriver systemet eller genom att rita upp systemet med hjälp av diagram. Modellen tolkas sedan med hjälp av en algoritm för att skapa testfallen. Algoritmens utseende varierar beroende på hur stor del av modellen som ska täckas av testfallen, vilka kriterier som ska uppfyllas (exempelvis kortaste vägen eller snabbaste vägen för att nå en nod i en modell) samt till vilken plattform testaren vill ha testfallen. Det ska i efterhand vara enkelt att gå in och ändra i modellen ifall något ändras i koden. Detta i sin tur gör det enkelt att testa ändringarna i det nya systemet då testfallen baseras på modellens utseende.

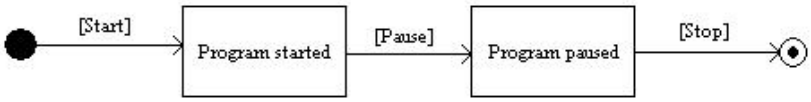
2.1.2 UML

För att modellera system används för det mesta UML [4 och 5], Unified Modeling Language. UML är ett modelleringsspråk som används inom mjukvaruutveckling men används även inom andra ämnen. I UML används ett antal grafiska objekt för att rita upp en abstrakt modell av ett system. Det finns flera olika sätt att modellera ett system i UML. Först och främst är ett system antingen statiskt eller dynamiskt. Exempel på statiska system är klassdiagram (class diagrams) och sammansatta strukturdiagram (composite structure diagrams) som fokuserar på den statiska strukturen hos ett system. De statiska systemen ritas upp genom en kombination av objekt, attribut, metoder och relationer. Sekvensdiagram (sequence diagrams), aktivitetsdiagram (activity diagram) och tillståndsdiagram (state machine diagrams) är exempel på dynamiska system som fokuserar på att visa det dynamiska beteendet hos ett system. Detta uppfylls genom att visa samverkan mellan olika objekt samt genom att visa ändringar i det interna tillståndet hos ett objekt.

I det modellbaserade testverktyg som ska tas fram kommer tillståndsdiagram [6] att användas för att rita upp modellerna. Tillståndsdiagram är en typ av diagram som används mest inom datorvetenskap och används där för att beskriva beteendet hos ett system. Det enda krav som ställs på ett system är att det innehåller ett begränsat antal tillstånd. Tillstånden i ett tillståndsdiagram länkas samman via olika händelser, vilket inom datavetenskap brukar kallas metodanrop. Ett stort system representeras vanligtvis av ett tillståndsdiagram som består av

flera små tillståndsdiagram. Vart och ett av dessa tillståndsdiagram representerar i sin tur ett objekt av en specifik klass och i detta diagram visas de olika tillstånden som objektet kan anta i systemet. För att kunna rita upp ett tillståndsdiagram används följande element:

- Fylld cirkel, representerar det initiella tillståndet.
- Ofylld cirkel med en mindre, fylld cirkel inuti, representerar det slutliga tillståndet.
- Rektangel med text i, representerar ett tillstånd som inte är det initiella eller slutliga tillståndet. Texten i rektangeln beskriver tillståndets namn.
- Streck med pil samt text vid, representerar ett tillståndsbyte. Pilen visar åt vilket håll tillståndsbytet sker och texten beskriver vilken händelse som orsakar tillståndsbytet.



Figur 2.1.2.1: Exempel på ett tillståndsdiagram.

2.1.3 Användbarhet

Något som tidigt iakttogs under examensarbetets upptakt var problemet med att mäta användbarhet på ett effektivt sätt. Användbarhet i dess grundform är något som upplevs olika från användare till användare och därför är det svårt att mäta på ett effektivt sätt. För att kunna ge en bra beskrivning av vad användbarhet är letades en ISO-standard upp. Enligt ISO 9241-11 (Gulliksen och Göransson, 2002, s.62) är användbarhet:

> *”Den utsträckning till vilken en specificerad användare kan använda en produkt till att uppnå specifika mål, med ändamålsenlighet, effektivitet och tillfredsställelse, i ett givet användningssammanhang.”*

Det finns givetvis flera tolkningar av användbarhet och en del författare har valt att dela upp användbarhet i flera mindre attribut. Definitionerna förenklar utvärderingen eftersom de mindre attributen är möjliga att mäta med enkla mätmetoder. Det är t.ex. lätt att mäta hur lång tid det tog att utföra en uppgift och hur många gånger en användare gjorde fel innan en uppgift blev utförd. Skulle du istället mäta användbarheten som ett enda attribut hade det varit svårare att dra några slutsatser om hur saker skulle kunna förbättras. Två av dessa tolkningar är gjorda av Nielsen [7] respektive Schneiderman [7] och dessa kan ses i tabell 2.1.3.1.

Tabell 2.1.3.1: Mätbara aspekter av användbarhet enligt ISO 9241-11, Nielsen och Schneiderman, omarbetat från Gulliksen & Göransson [7].

ISO 9241-11	**Nielsen**	**Schneiderman**
Effektivitet	Effektivt att använda Lätt att lära	Tid att utföra en uppgift Tid att lära
Ändamålsenlighet	Lätt att komma ihåg Få fel	Kvarhållande i minnet över tid Frekvensen användarfel
Tillfredsställelse	Subjektivt tilltalande	Självupplevd tillfredställelse

I grund och botten är dessa båda tolkningar identiska men i fortsättningen kommer Schneidermans tolkning att användas för att mäta användbarhet. Utifrån Schneidermans användbarhetsdefinition var det möjligt att sätta upp mätbara användbarhetsmål för användargränssnittet. Användbarhetsmått som låg till grund för användbarhetsmålen var:

- Tiden för att slutföra en specifik uppgift (effektivitet).
- Tiden som spenderas på att rätta fel som uppkommer (effektivitet).
- Tiden för att lära sig funktionalitet som har samma design (effektivitet).
- Antal användarfel (ändamålsenlighet).
- Andelen uppgifter som slutförs framgångsrikt vid första försöket (ändamålsenlighet).
- Bedömningsnivå av tillfredsställelsen (tillfredsställelse).
- Bedömningsnivå av felhanteringen (tillfredsställelse).
- Frekvensen av återanvändning (tillfredsställelse).

Förutom dessa användbarhetsmål måste användarens mål och intentioner specificeras och en beskrivning av användningssammanhanget, användarna, uppgifterna, utrustningen samt miljöerna ges. För att uppnå användbarheten som önskades i användargränssnittet bestämdes ett antal riktlinjer och designprinciper som skulle följas. Designprinciperna som valdes har specificerats av Donald A. Norman [8]. Dessa användes för att säkerställa att de första prototypernas användargränssnitt blev någorlunda användarvänliga och effektiva att använda. Designprinciperna är:

- *Konceptuella modeller (conceptual models)*. En konceptuell modell gör det möjligt för användaren att förutse resultatet av sina handlingar och tillåter denne att simulera det mentalt. Designern ska med hjälp av användargränssnittet tydligt kunna klargöra för användaren vad som förväntas av en funktion. Ett exempel är att låta en person ställa in värmen i sin bil. Personen väljer via användargränssnittet att temperaturen ska ändras till 22 grader och förväntar sig då självklart att temperaturen i bilen ska rätta sig efter detta. Om temperaturen skulle bli något annat än 22 grader hade det varit en dåligt designad konceptuell modell eftersom användaren förväntade sig ett annat resultat.

- *Återkoppling (feedback)*. Återkoppling ger användaren kontinuerlig information om vad som händer och som har hänt. Detta behövs för att användaren aldrig ska hamna i en situation som gör henne osäker eller obekväm. Ett exempel på brist på återkoppling är om en användare trycker på en knapp och ingenting händer. Användaren blir då osäker på om knappen trycktes in tillräckligt hårt och trycker en gång till, vilket inte ska behövas. Knappen trycktes med största sannolikhet in tillräckligt hårt redan första gången, enda problemet var att användaren inte fick någon information om att det lyckades.
- *Restriktioner* (*constraints*). Restriktioner gör det omöjligt för användaren att utföra något annat än det tillåtna. Detta är den mest säkra metoden för att få någonting till att fungera som det är tänkt. Ett varningsmeddelande om att något är förbjudet är ett exempel på när restriktioner inte använts på ett bra sätt i designen. Det ska inte vara möjligt att utföra något som är förbjudet om man följt designprincipen för restriktioner. Ett exempel på bra design är en batteribehållare som designats på ett sådant sätt att det endast är möjligt att sätta i dem på ett sätt. Det är uppenbart för användaren hur batterierna ska sättas i utan att behöva läsa ett långt varningsmeddelande.
- *Självinstruerande* (*affordances*). Självintruerande är att göra det uppenbart för användaren vilka handlingar som är lämpliga och olämpliga att använda i ett speciellt skede. Exempelvis kan en designer använda sig av röd varningsfärg för att tydligt visa användaren att en viss funktion inte bör användas. En annan metod hade varit att använda sig av grön färg för att visa att en funktion är lämplig att användas. Ett vanligt exempel på användning av metoden självinstruerande är när en utvecklare väljer att göra knappar och menyalternativ ljust gråa, vilket berättar för användaren att de för tillfället inte är möjliga att använda.

Utöver dessa designprinciper bestämdes det även att Schneidermans åtta gyllene regler [7] för design skulle has i åtanke under designen. En del av dem är redan nämnda i ovanstående designprinciper av Norman. Reglerna är:

1. Sträva efter konsekvens.
2. Möjliggör för frekventa användare att använda genvägar.
3. Erbjud informativ återkoppling.
4. Designa dialoger som bidrar till närhet.
5. Erbjud avvärjande av fel och enkel felhantering.
6. Tillåt enkla sätt att gå bakåt i handlingar.
7. Stöd den inre känslan av kontroll.
8. Reducera belastningen på korttidsminnet.

2.2 Utvärdering av dagens gränssnittsstandard för modell- och testverktyg

För att få en uppfattning om dagens gränssnittsstandard för modell- och testverktyg så utfördes en undersökning av ett antal program. Först utvärderades fyra test- och modellprogram och efter det utvärderades även fyra program som inte var modell- eller testprogram. Programmen för utvärderingen valdes ut tillsammans med System Verification. Analysen är utförd på egen hand och är enbart baserad på mina egna tankar och värderingar. Det som ligger i granskningens fokus är programmens användbarhet och detta beskrivs med de termer som presenterades i kapitel 2.1.3. Programmen har testats utan användarmanual i så stor utsträckning som möjligt för att på ett rättvist sätt få en uppfattning om deras användbarhet.

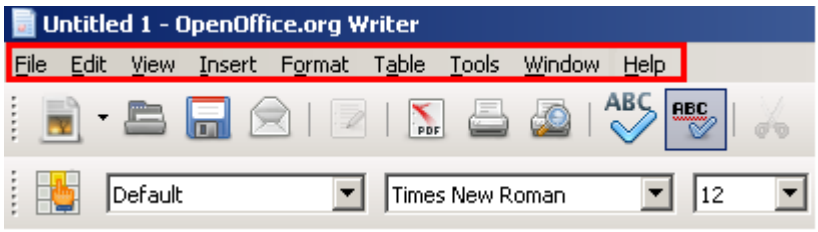
2.2.1 Terminologi

För att inga missförstånd ska uppstå kommer här en kort terminologi med sådant som kan hittas i användargränssnitt. Detta för att läsaren ska förstå vilken del av användargränssnittet det pratas om.

1. *Ram*, fältet längst upp, vanligtvis blått, på ett program där namnet på programmet står samt ibland även den öppna filens namn.

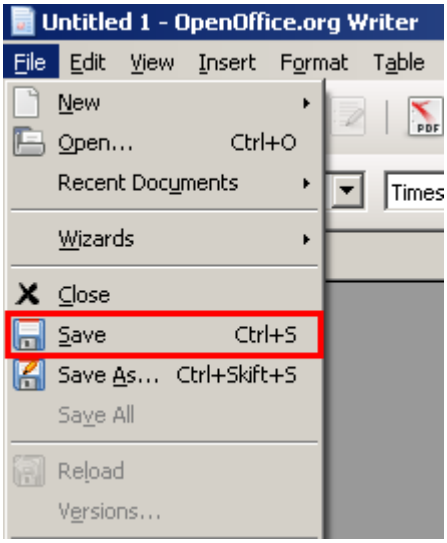
Figur 2.2.1.1: Exempel på en ram, markerad med en röd rektangel, i OpenOffice.org Writer.

2. *Meny*, precis under programmets ram finns menyn där bland annat kommandon för att spara och öppna filer finns att tillgå.



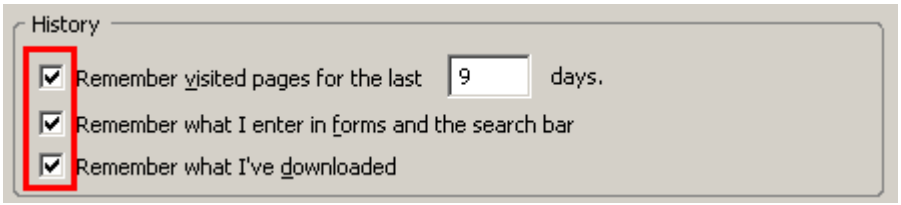
Figur 2.2.1.2: Exempel på en meny, markerad med en röd rektangel, i OpenOffice.org Writer.

3. *Menyalternativ*, ett av alternativen som kan väljas i en meny.



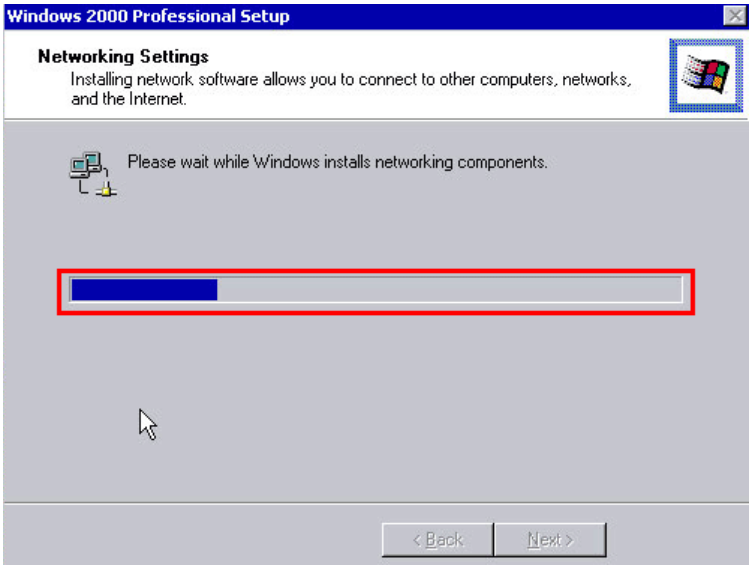
Figur 2.2.1.3: Ett menyalternativ, markerad med en röd rektangel, i OpenOffice.org Writer.

4. *Kryssruta*, låter dig bocka i något som du vill sätta igång eller bocka av något du inte längre vill ha.



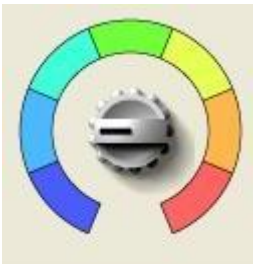
Figur 2.2.1.4: Exempel på kryssrutor, markerad med en röd rektangel, i Mozilla Firefox.

5. *Progressbar*, en mätare som visar hur ett förlopp fortskrider. Exempelvis så visas det hur långt en installation har hunnit.



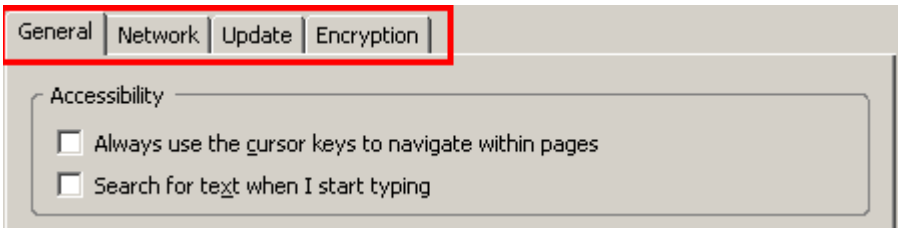
Figur 2.2.1.5: Exempel på en progressbar, markerad med en röd rektangel, i Windows 2000.

6. *Ratt*, något som det går att snurra på för att ställa in ett önskat värde.



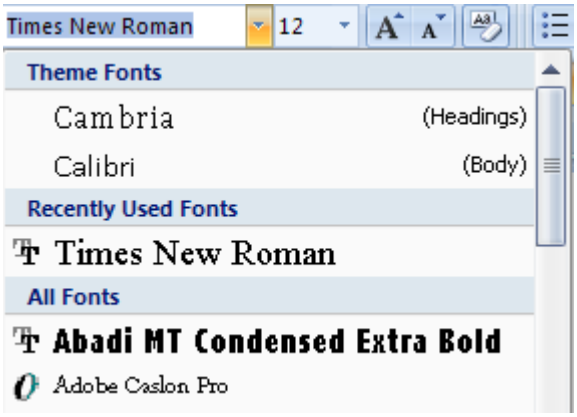
Figur 2.2.1.6: Exempel på en ratt i Qtronics.

7. *Fliksystem* och *flikar*, ett sätt att bläddra mellan olika vyer i ett program. Fungerar på samma sätt som en pärm.



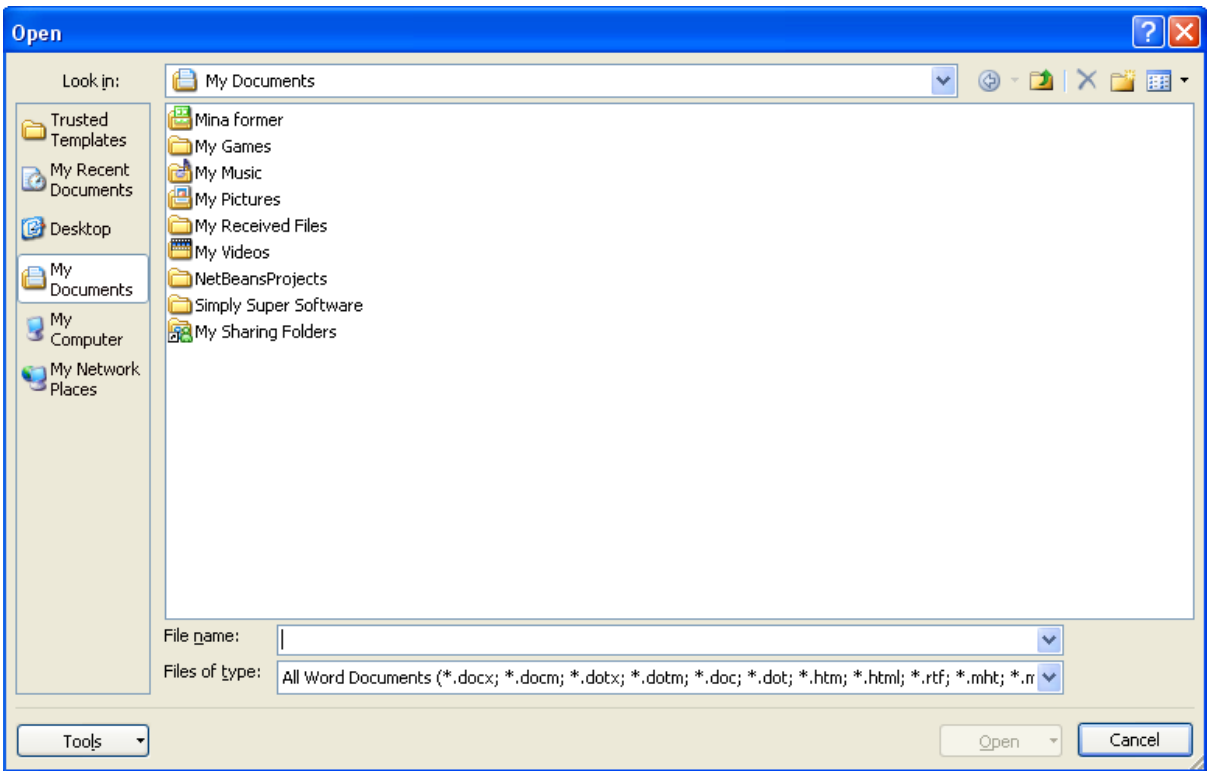
Figur 2.2.1.7: Exempel på ett fliksystem, markerad med en röd rektangel, i Mozilla Firefox.

8. *Rullista*, en lista med alternativ som det endast går att välja ett ur. Det är möjligt att ändra valet genom att rulla på hjulet på musen utan att hela listan visas.



Figur 2.2.1.8: Exempel på en rullista i Microsoft Office Word 2007.

9. *Fildialog*, en vy där det går att hitta en specifik fil, t.ex. för att öppna den.



Figur 2.2.1.9: Exempel på en fildialog i Microsoft Office Word 2007 för att öppna en fil.

10. *Tooltip*, en förklarande text som dyker upp om användaren sätter muspekaren över en knapp en längre tid.

Figur 2.2.1.10: Exempel på ett tooltip i Microsoft Windows XP.

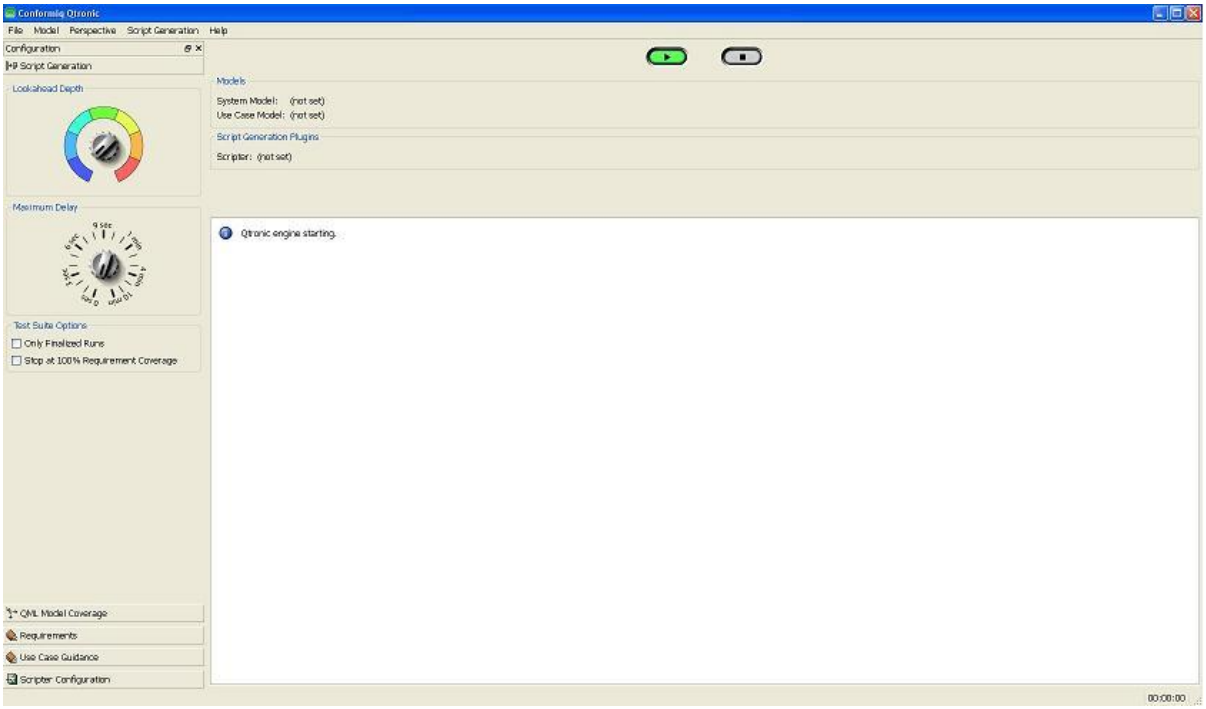
2.2.2 Utvärdering av dagens test- och modelleringsprogram

De modell- och testverktyg som utvärderades var:

- Qtronics, version 1.3.4, från Conformiq [9]
- ADT, Automatic Test Designer, version 6.0.1 (Evaluation), från AtYourSideConsulting [10]
- Visual Paradigm for UML, version 6.4, från Visual Paradigm [11]
- Microsoft Office Visio 2007 [12]

2.2.2.1 Utvärdering av Qtronics

Företaget Conformiq har utvecklat ett testverktyg som heter Qtronics. Med hjälp av en modell, som ritats i ett annat program, kan ett antal testfall genereras i detta program. Det går alltså ej att skapa modeller eller att exekvera testfall i det här programmet. När programmet startas upp ser det ut enligt bilden nedan:



Figur 2.2.2.1.1: Startbild från Qtronics.

Det första som läggs märke till är ett stort informationsfönster, enligt mig lite för stort, som ger en information om vad programmet har gjort. Utöver detta syns det relativt tydligt att både modell och scripter inte blivit specificerade. Trots detta ser det ändå ut som att testgenereringen kan startas då startknappen är grön. Knappen borde istället varit samma färg som stoppknappen, dvs. grå så att användaren inte tror att det går att starta testgenereringen. Start- och stoppknapparna är välgjorda då de hämtat ikonerna från vardagliga apparater, som t.ex. DVD-spelare, vilket gör att användaren har en klar konceptuell modell angående vad knapparna är till för. Dock borde Conformiq placerat knapparna under Models- och ”Script Generation Plugins”-fälten så att det hela följde ett uppifrån-och-ner-perspektiv, dvs. att

användaren börjar uppifrån och arbetar sig neråt. En annan positiv sak hos Qtronics är att de olika fälten tydligt avgränsas med gråa linjer, kolla t.ex. avgränsningen mellan fälten för modell och scripter i figur 2.2.2.1.1. Med hjälp av dessa linjer kan liknande funktionalitet samlas så att inte allting läggs på samma ställe. Vill användaren hitta något som har med modeller att göra kollar hon självfallet i modellfältet.

Innan det är möjligt att starta testgenereringen måste alltså modell och scripter väljas. I figur 2.2.2.1.1 syns det att det inte finns några knappar för att göra detta. Istället måste användaren gå via menyn för att sätta dessa båda. Det går också att använda sig av knappkombinationer, ”Ctrl+M” för att sätta modell och ”Ctrl+X” för att sätta scripter fast det är knappast något som en helt ny användare hade använt sig av. Givetvis är dessa knappkombinationer utmärkta för de mer erfarna och avancerade användarna även om ”Ctrl+X” är synonymt med att klippa ut något. När en modell eller scripter valts uppdateras informationen, dvs. det står inte längre att de inte blivit satta, och en informativ textrad skrivs ut i informationsfönstret. Denna återkoppling hjälper användaren att förstå vad som händer baserat på hennes val och gör att hon inte behöver känna sig obekväm. Alla har säkert någon gång testat ett program där ingen återkoppling alls givits förrän långt efter att en funktion använts. Användaren blir då väldigt osäker på om hon verkligen lyckats genomföra det önskade och trycker kanske på knappen mer än en gång. Det kan också gå så långt att användaren väljer att stänga ner programmet i tron att programmet hängt sig. Något som däremot är mindre bra är att det saknas knappar för att sätta modell och scripter. Eftersom dessa funktioner används varje gång som testfall ska genereras hade det varit bra om det funnits knappar åt dessa grundläggande funktioner.

Något annat som också är dåligt designat i Qtronics är fliksystemet som används i konfigurationsfönstret. Istället för att ha alla flikar samlade på en sida så hamnar de både i över- och underkanten av konfigurationsfönstret. Detta leder till att muspekaren måste flyttas onödigt långa sträckor mellan de olika flikarna. Kolla bilden 2.2.2.1.2 till höger för att få en uppfattning.

Figur 2.2.2.1.2:
Fliksystemet i Qtronics.

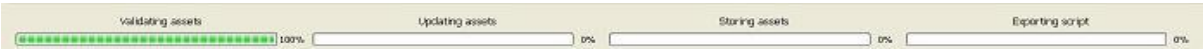
En annan funktion i Qtronics är rattar som återfinnes i konfigurationsfliken. Rattkonceptet är enligt mig en dålig idé då användaren lätt kan råka dra muspekaren från den ena sidan till den andra och på så sätt lyckas öka eller minska värdet med cirka 50 procent. Det är också svårt att göra små justeringar eftersom muspekaren måste röras både i höjd- och sidled vilket gör att det krävs mer precision från användarens sida. Utöver detta så bör rattarna inte ha små cirkelformade taggar som sticker ut. Dessa lurar användaren och kan få denna till att tro att ett annat värde än det riktiga är satt. En bild på rattarna finns i figur 2.2.2.1.1.

Innan användaren väljer att generera testfall kan en del inställningar göras som påverkar hur stor del av modellen som kommer att täckas av testfallen. För att komma åt dessa inställningar måste fliken ”QML Model Coverage” väljas. Den flik som dyker upp är enligt mig väldigt lätt att förstå, klickar användaren t.ex. i kryssrutan med texten ”State Coverage” så

betyder det att tillstånden i modellen kommer att täckas av testfallen som genereras. Likaså är det självklart att de täckningsalternativ som har ej markerade kryssrutor inte kommer att användas vid testfallsgenereringen. Detta är ett exempel på en lyckad konceptuell modell. En bild på detta återfinns i figur 2.2.2.1.3 nedan:

Figur 2.2.2.1.3: Sätta modelltäckning i Qtronics.

Efter att startknappen tryckts ner och testfallen börjar genereras märker användaren att det finns mer än en progressbar. Istället för en enda progressbar som har text under sig som beskriver vad som sker just nu, har Qtronics istället hela fem stycken progressbars. Fyra av dessa visar individuella förlopp och den femte visar det övergripande förloppet. Det blir onödigt mycket information som visas på en och samma gång och användaren får således svårt att få en klar överblick över vad som händer. Återkopplingen med progressbars i sig är en väldigt bra tanke men att använda sig av så många som fem är överdrivet.

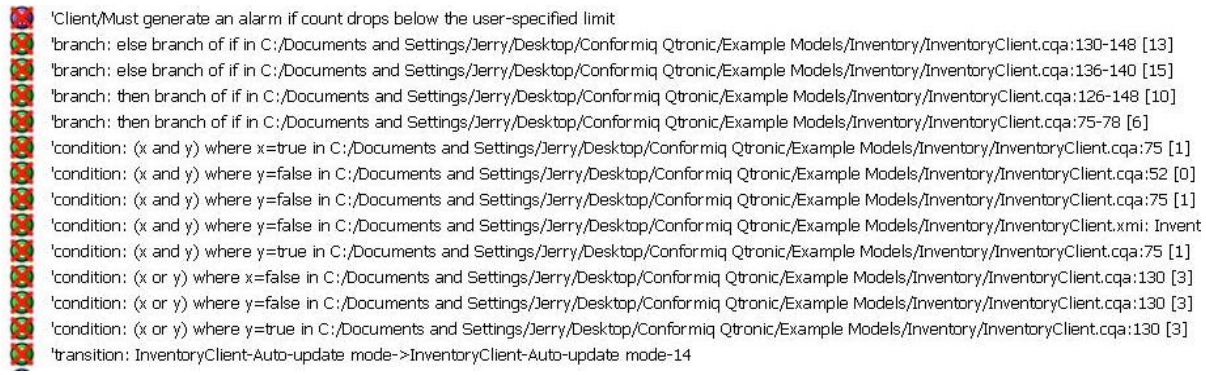


Figur 2.2.2.1.4: De fyra individuella progressbarsen i Qtronics.

Under testgenereringen kommer information att skrivas ut i informationsfönstret, vilket är positivt då det hela tiden ges information om hur testgenereringen fortgår. Användaren får en tydlig återkoppling om vad det är som sker. Något som inte visas i någon figur men som visas i programmet är den totala tiden det tog för att generera testfallen. Detta kan vara väldigt bra att veta för framtida testfallsgenereringar om bara något litet har ändrats i modellen. Användaren har kanske en deadline som hon inte får missa och då kan det vara bra att veta hur lång tid den nya testfallsgenereringen kommer att ta.

Figur 2.2.2.1.5: Övergripande progressbar i Qtronics samt återkoppling från felfri testgenerering.

Något som upptäcktes under genereringen av testfallen var att det kanske skulle vara önskvärt att ha en pausknapp. Denna knapp skulle varit bra att ha ifall en användare vill vara närvarande under hela testgenereringsförloppet. Kanske blev hon tvungen att gå ifrån för att gå på ett möte eller liknande och då är det bra att ha möjlighet att pausa testgenereringen. Om testgenereringen stoppas eller om något går fel under förloppet så visas detta tydligt i informationsfönstret. De vanligtvis gröna eller blå informationsikonerna får ett rött kryss över sig vilket är en tydlig varning för användaren om att något gått fel. Hur detta ser ut kan du se i figur 2.2.2.1.6 nedan:

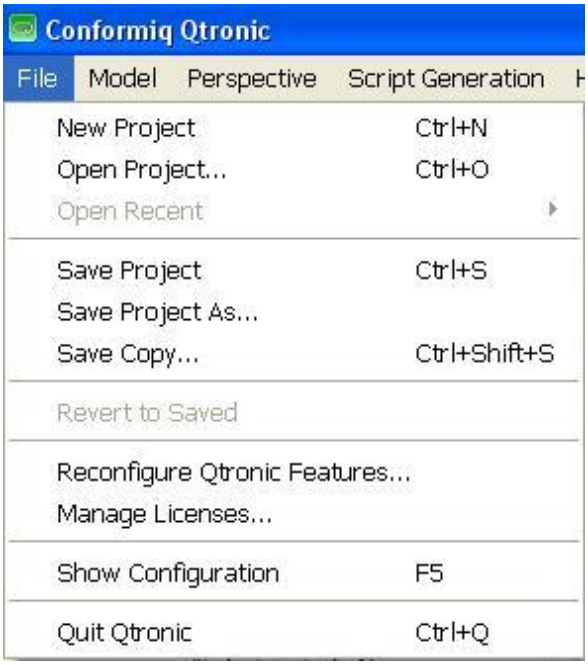


Figur 2.2.2.1.6: Återkoppling från felaktig testgenerering i Qtronics.

Något som var väldigt svårt att genomföra var att välja en scripter. Att hitta själva fildialogen var inte svårt men att hitta rätt fil visade sig vara ett mindre projekt. För det första så startade fildialogen i grundkatalogen för installationen och inte i katalogen där alla scripters fanns. Vidare så listade Qtronics alla *.dll-, *.class- och *.jar-filer som fanns så även efter att ha hittat rätt katalog så visades en massa *.dll-filer som inte var scripters. För att undvika detta problem borde användaren hamna i rätt katalog direkt och det borde finnas ett filter som gör att enbart scripters visas. Utvecklarna borde ha använt sig av restriktioner på ett bättre sätt i det här fallet genom att göra det omöjligt för användaren att välja *.dll-filer som inte var scripters.

I menyerna syns det att utvecklarna gjort ett bra jobb genom att använda avgränsande streck som delar in liknande funktionalitet i grupper. De använder sig också av grå text i menyerna

för att synliggöra restriktioner, detta ger användaren information om vilken funktionalitet som går och inte går att använda för tillfället. Det är för övrigt också bra att de använder sig utav standardiserade knappkombinationer såsom ”Ctrl+S” som motsvarar spara och ”Ctrl+N” som motsvarar att skapa något nytt. Funktionerna som finns i filmenyn, se figur 2.2.2.1.7, är väl indelade men en del funktioner hör inte hemma i denna meny. Alternativ som ”Show Configuration”, ”Reconfigure Qtronic Features...” och ”Manage Licenses...” borde finnas i en helt annan meny, som förslagsvis borde heta ”Settings”. Sen kan det också diskuteras om en del funktioner är nödvändiga eller inte. Vad skiljer t.ex. ”Save Project As...” från ”Save Copy...”?



Figur 2.2.2.1.7: Återkoppling från felaktig testgenerering i Qtronics.

Positiva lärdomar att ta med sig:

- Knappar för att starta eller stoppa något är väldigt informativa om de använder sig av ikoner som är hämtade från apparater som används i det vardagliga livet.
- Kryssrutor är ett koncept som är lätt för en användare att förstå förutsatt att beskrivningarna är bra.
- Ge användaren kontinuerligt återkoppling annars kan det lätt uppfattas som att programmet hängt sig eller att en funktion inte satts igång. I programmet fick användaren information via informationsfönstret och progressbarsen.
- Använd färgkodning för att visa vad som gått rätt och vad som gått fel. Grönt visar att något gått bra och rött att något inte gått som förväntat.
- Sortera liknande funktionalitet med hjälp av tydliga avgränsningar, t.ex. linjer, så att användaren snabbt kan hitta det hon söker.

- Bra att tiden för testgenereringen visas ifall användaren vill ändra något smått i en modell och sedan generera om testfall. Ger användaren en bra uppfattning om hur mycket tid som krävs ifall hon har en deadline att passa.
- Det gick att se vad som inte var satt för tillfället och det gjorde att användaren kunde avgöra vad som behövde göras innan testgenereringen kunde startas. Kanske borde det markerats på ett tydligare sätt än med text.
- I grund och botten var det enkelt att generera testfall. Det krävdes endast att användaren gjorde två inställningar innan testgenereringen kunde startas. En tumregel att följa sen är att försöka hålla det så enkelt som möjligt.

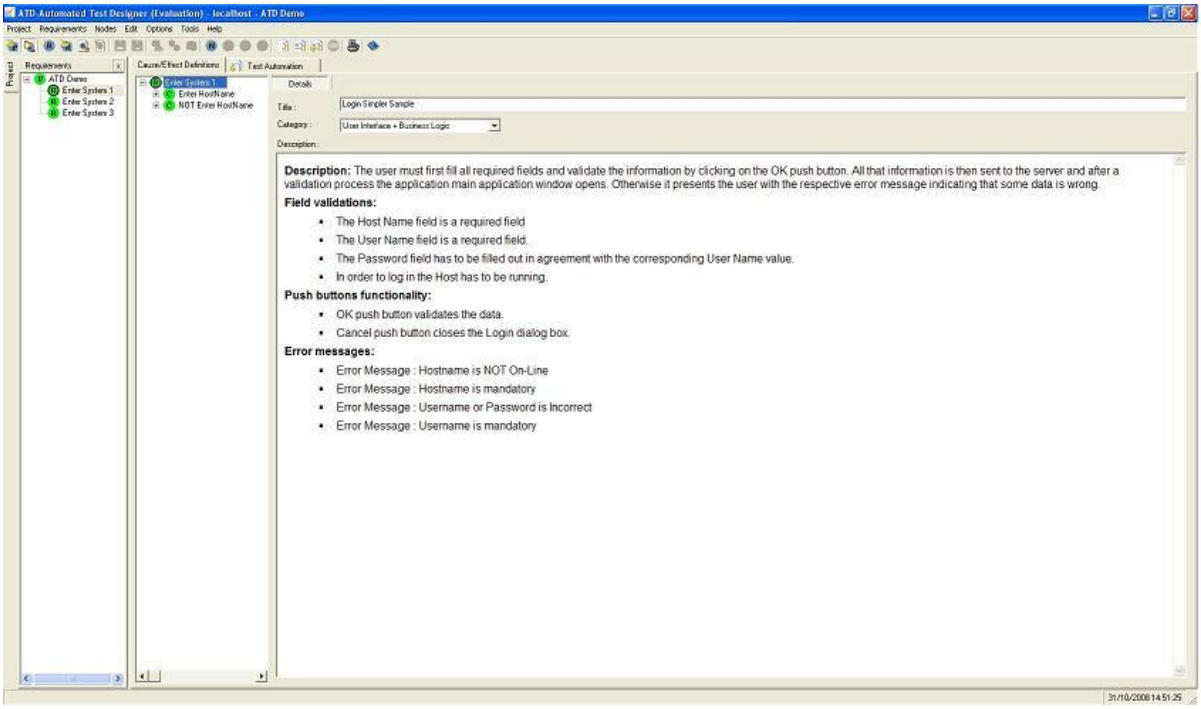
Negativt som bör undvikas:

- Placera inte fält i en, för uppgiften, onaturlig följd. Gör det istället lätt för användaren att på ett naturligt sätt arbeta sig igenom något, antingen uppifrån och ner eller från vänster till höger. Om användaren måste göra inställningar för att kunna starta något ska dessa ligga före startknappen.
- De mest använda funktionerna bör få knappar. Användaren ska inte behöva gå via en meny varenda gång som hon vill använda en flitigt använd funktion.
- Informationsfönstret tar upp alldeles för stor plats och fångar användarens fokus. Försök att göra det precis lagom stort, inte för litet och inte för stort.
- Undvik onödigt långa avstånd mellan flikar då detta ökar tiden för att flytta musen från en flik till en annan.
- Använd endast en progressbar för att visa hur testgenerering eller dylikt fortskrider. Om det är flera saker som sker kan en förklarande text skrivas ut undertill.
- Rattar är en god idé men funkar inte vidare bra i datorsammanhang och bör undvikas. Det kan hända att muspekaren råkar dras för mycket åt ett håll och således ökas eller minskas värdet med ca 50 % istället för den önskade justeringen.
- När en fil ska öppnas bör filerna filtreras så att endast det som kan öppnas ska synas. Ett exempel är om en *.doc-fil ska öppnas så visas endast *.doc-filer. Om möjligt ska fildialogen även startas i rätt katalog
- Använd inte välkända knappkombinationer för annan funktionalitet. T.ex. bör inte ”Ctrl+X” användas till något annat än klippa ut.
- Startknappen var grönfärgad även när testgenereringen inte gick att starta. Den gröna färgen lurade användaren till att tro att knappen gick att använda. Istället bör de funktioner som ej går att användas för tillfället gråfärgas för att indikera att de är inaktiva.
- Det fanns ingen möjlighet att pausa testgenereringen. Ett enkelt sätt att lösa detta vore att lägga till en pausknapp mellan start- och stoppknappen.

- Sortera funktionalitet som hör samman i grupper. Samla inte två olika gruppers funktionalitet i samma meny, ett exempel på detta är filmenyn i Qtronics som har en del funktioner från inställningsmenyn.

2.2.2.2 Utvärdering av ADT, Automatic Test Designer

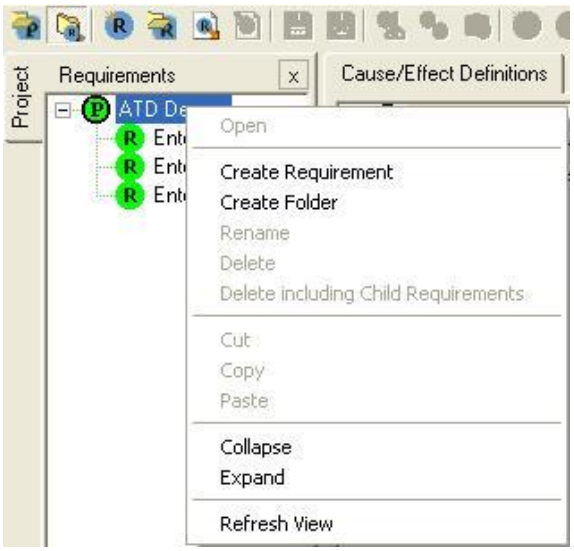
Från företaget AtYourSideConsulting kommer programmet ADT, Automatic Test Designer. I ADT kan en användare specificera ett system samt kraven på detta system och utifrån dessa krav genereras sedan testfallen. Innan själva klienten går att starta måste användaren eller den lokala nätadministratören sätta upp en server att köra mot. Efter att ha anslutit sig till en server ska ett projekt väljas och detta sker via en rullista som dyker upp i ett nytt fönster. Personlig känns det lite omotiverat att använda en rullista då det antagligen inte finns särskilt många projekt att välja mellan. Istället hade programmet kunnat visa de fem senaste projekten som användaren arbetat med och även gett möjlighet att öppna projekt via en fildialog. I evalueringsversionen av ADT kan endast ”ADT Demo” väljas och det är valt redan från början. Det går alltså inte att skapa ett helt nytt projekt från denna dialogruta. Inte heller går det att skapa ett nytt projekt på annat sätt i programmet, förhoppningsvis är detta en begränsning hos utvärderingsversionen. Efter att ha valt projekt slussas användaren vidare till startsidan för programmet.



Figur 2.2.2.2.1: Startbild från ADT.

Längst till vänster i programmet finns en vy där projektet och kraven för projektet presenteras. Projektet identifieras via en grön cirkel med ett ”P” i och kraven via gröna cirklar med ”R” i. Detta gör att användaren relativt tydligt kan se vilka som är projekt och vilka som är krav. Det hade dock varit ännu lättare för användaren att se skillnaden om projektet och kraven dessutom haft olika färger. Utöver detta ser användaren även att projektet fungerar som en katalog där de underliggande kraven samlas. Dubbelklickar användaren på ett av kraven dyker det upp till höger i ”Cause/Effect Definitions”-fliken och en beskrivning av kravet ges. Det som står i beskrivningen är det som sedan ligger som grund för vilka testfall

som kommer att genereras. I samma fliksystem finns även en flik som heter ”Test Automation”. I denna flik kan automatiseringsverktyget med vilket testfallen ska genereras väljas och det går även att se inställningarna för kravet som är valt. Det är positivt att dessa inställningar inte visas på samma gång som det som visades i fliken ”Cause/Effect Definitions” eftersom det med största sannolikhet hade blivit för mycket att hantera för en oerfaren användare.



Figur 2.2.2.2.2: Två av sätten att skapa ett nytt krav i ADT.

Det finns tre olika sätt att skapa ett nytt krav. Två av sätten visas i figuren ovan. Antingen högerklickar användaren på projektet och väljer ”Create Requirement” eller så klickar användaren på knappen som har en ikon med en blå cirkel med en stjärna uppe i vänstra hörnet samt ett ”R” i. Stjärnan är upp i vänstra hörnet symboliserar att något nytt skapas och används i flera andra program. Det tredje sättet är att gå via menyalternativet ”Requirements” och där välja ”Create Requirement”. Alla tre sätten är enkla att upptäcka för användaren även om menyalternativet kommer att användas minst efter att användaren blivit van vid programmet. Vyn för det nya kravet öppnas inte efter att det skapats utan istället är det krav som senast beskådades fortfarande synligt. För att komma till det nya kravet måste användaren dubbelklicka på det. Om en användare skapar ett nytt krav vill hon med största sannolikhet arbeta med just det så vyn borde ändras till det nya kravet.

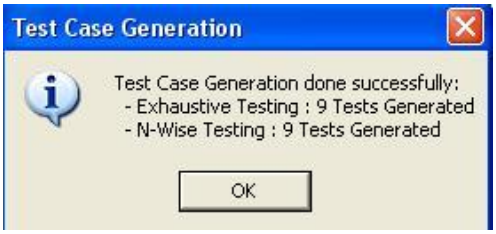
Figur 2.2.2.2.3: Ikonernas utseende efter att en nod markerats i ADT.

I figuren ovan syns knapparna i ADT efter att en nod markerats. De olika knapparna är indelade i olika grupper med hjälp av avgränsande streck och knappar som inte går att använda för tillfället är gråmarkerade. Tar vi en titt på ikonerna ser vi att det använts en del standardikoner. Ikonerna för att öppna ett projekt eller ett krav använder en öppen mapp kombinerat med antingen ett ”P” eller ett ”R” beroende på vad som ska öppnas. Det är också lätt att veta vilka knappar som används för att klistra ut, kopiera eller klistra in något eftersom även de använder standardikoner ifrån andra program. För att hantera noder finns det fyra olika knappar. Den första använder en ikon som bygger på samma princip som ikonen för att skapa ett nytt krav. Enda skillnaden är att ikonen har ett ”N” i sig istället för ett ”R”. De tre resterande knapparnas funktion är från vänster till höger:

- Ta bort en nod.
- Flytta en nod uppåt.
- Flytta en nod neråt.

Ta bort en nod representeras av en blå cirkel med ett ”N” i samt med ett kryss över. Detta visar klart och tydligt för användaren att något ska raderas. De två sista knapparna används för att flytta upp eller ner en nod inom ett krav. Ikonerna för dessa funktioner är även de bra konstruerade, en pil uppåt i en blå cirkel markerar att en nod kan flyttas uppåt och en blå cirkel med en pil riktad neråt markerar att en nod kan flyttas neråt. Varför just färgen blå använts till cirklarna i ikonerna är konstigt eftersom cirklarna är gröna för projekt, krav och noder.

I figur 2.2.2.2.3 kan även knappen för att starta testgenereringen ses till höger om knapparna för att hantera noder. Ikonen för denna funktion representeras av två dokument som ligger ovanpå varandra och som har en bock över sig. För mig personligen var det inte helt självklart vad denna knapp gjorde. Under tiden som testfallen skapas visas en progressbar i programmets nedre kant vilket tydligt ger användaren återkoppling om hur testgenereringen fortlöper. Efter att alla testfall skapats visas ett informationsmeddelande. Ett exempel på hur detta kan se ut finns i figur 2.2.2.2.4 nedan. Det är mycket bra att användaren får denna återkoppling eftersom det hjälper användaren att se vad som skapats.

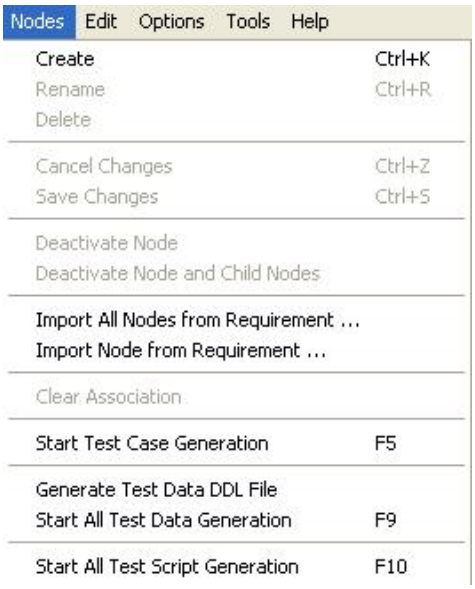


Figur 2.2.2.2.4: Informationsmeddelande efter testgenerering i ADT.

I ADT går det att stoppa testfallsgenereringen genom att trycka på en stoppknapp. Ikonen för denna knapp ser ut som en stoppskylt och det är därför uppenbart för användaren att denna knapp stoppar något eftersom vi känner igen det från den riktiga världen. Att det är just testgenerering som stoppas förstår användaren eftersom den är grupperad tillsammans med knapparna för att starta testgenereringen. Det hela kan beskådas i figur 2.2.2.2.5 nedan. Något som bör påpekas här är att det var väldigt svårt att se någon skillnad på de tre knapparna för testgenereringen. Alla tre såg ganska snarlika ut och detta ledde till att användaren hade svårt att förstå vad de egentligen utförde. Vad var egentligen skillnaden i funktionalitet mellan dessa tre?

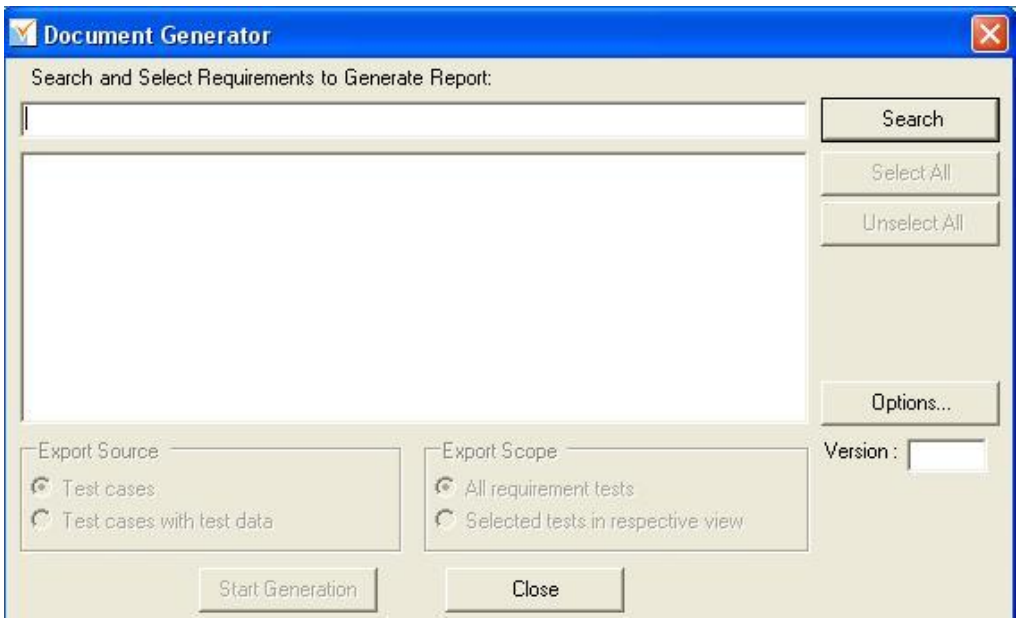
Figur 2.2.2.2.5: Stoppknappen i ADT när en testgenerering körs.

Något som också bör nämnas var att ett en del menyalternativ var placerade i fel menyer och borde placerats om för att användaren lättare skulle hitta dem. Låt oss ta en titt på ”Nodes”-menyn som finns att beskåda i figur 2.2.2.2.6. Vad gör alla testgenereringsalternativ i denna meny? Borde inte dessa istället placerats i en egen meny med namnet ”Test Generation”?



Figur 2.2.2.2.6: Menyn ”Nodes” i ADT.

En detalj som är viktig är att när en användare vill ta bort ett krav så frågar programmet om användaren verkligen vill ta bort det. Det är annars lätt hänt att ett krav tas bort av misstag. Försöker användaren att ta bort ett krav som har ett antal underkrav visas ett felmeddelande. Detta felmeddelande visas eftersom det finns en speciell funktion för att ta bort krav som har underkrav. Funktionen för att göra detta nås genom att högerklicka på kravet och sedan välja ”Delete including Child Requirements”. Det känns väldigt omotiverat att använda två olika funktioner för att ta bort krav. Istället borde det bara finnas en funktion för att radera krav. Denna funktion skulle då även radera underkraven till det krav som önskades raderas.



Figur 2.2.2.2.7: Fönstret som öppnas när en rapport ska genereras i ADT.

I figuren ovan ses fönstret som visas när en rapport ska genereras. För att kunna generera en rapport måste användaren först hitta kraven som hon vill ha och lägga till dem i rapporten. För att välja vilket format en rapport ska genereras i får användaren först trycka på

”Options..”-knappen i figur 2.2.2.2.7. Efter att det gjorts dyker ett nytt fönster upp och där kan användaren välja i vilket format rapporten ska genereras ifrån en rullista. I fönstret går det även att välja var rapporten ska skapas. En smidigare lösning hade varit att det fanns en knapp för att generera en rapport. Det projekt eller krav som användaren hade markerat innan hon tryckte på knappen är det som det ska skapas en rapport för. När knappen tryckts ska det öppnas en fildialog där användaren lätt kan välja var och i vilket format rapporten ska skapas. En sista sak som bör nämnas är att det var väldigt svårt att som användare få en övergripande förståelse av hur systemet som skapades såg ut i ADT. Det fanns ingen överblick eller liknande som det var möjligt att se hur allting var länkat i.

Positiva lärdomar att ta med sig:

- Använd konfirmationsdialoger för att fråga om något verkligen ska raderas så att inget raderas av misstag.
- Tydligt visa vilka funktioner som inte går att använda för tillfället genom att inaktivera knappar eller dylikt. Detta görs smidigast genom att gråmarkera de knappar som inte ska gå att användas.
- Ikoner ska i så stor utsträckning som möjligt baseras på standardikoner eller bilder som lätt kan associeras med funktionen via konceptuella modeller.
- Knappar med liknande funktionalitet ska grupperas tillsammans.
- Ge användaren kontinuerligt återkoppling om hur en testgenerering fortlöper. Använd t.ex. en progressbar för att visa hur stor del av genereringen som är avklarad. När allt är färdigt kan ett meddelande ges om att testgenereringen är avslutad.
- Grafiskt visa skillnad mellan ett projekt, krav och en nod. Detta gör det lättare för användaren att avgöra vad som är vad. I ADT borde det kanske markerats med färg också istället för att bara använda olika bokstäver.

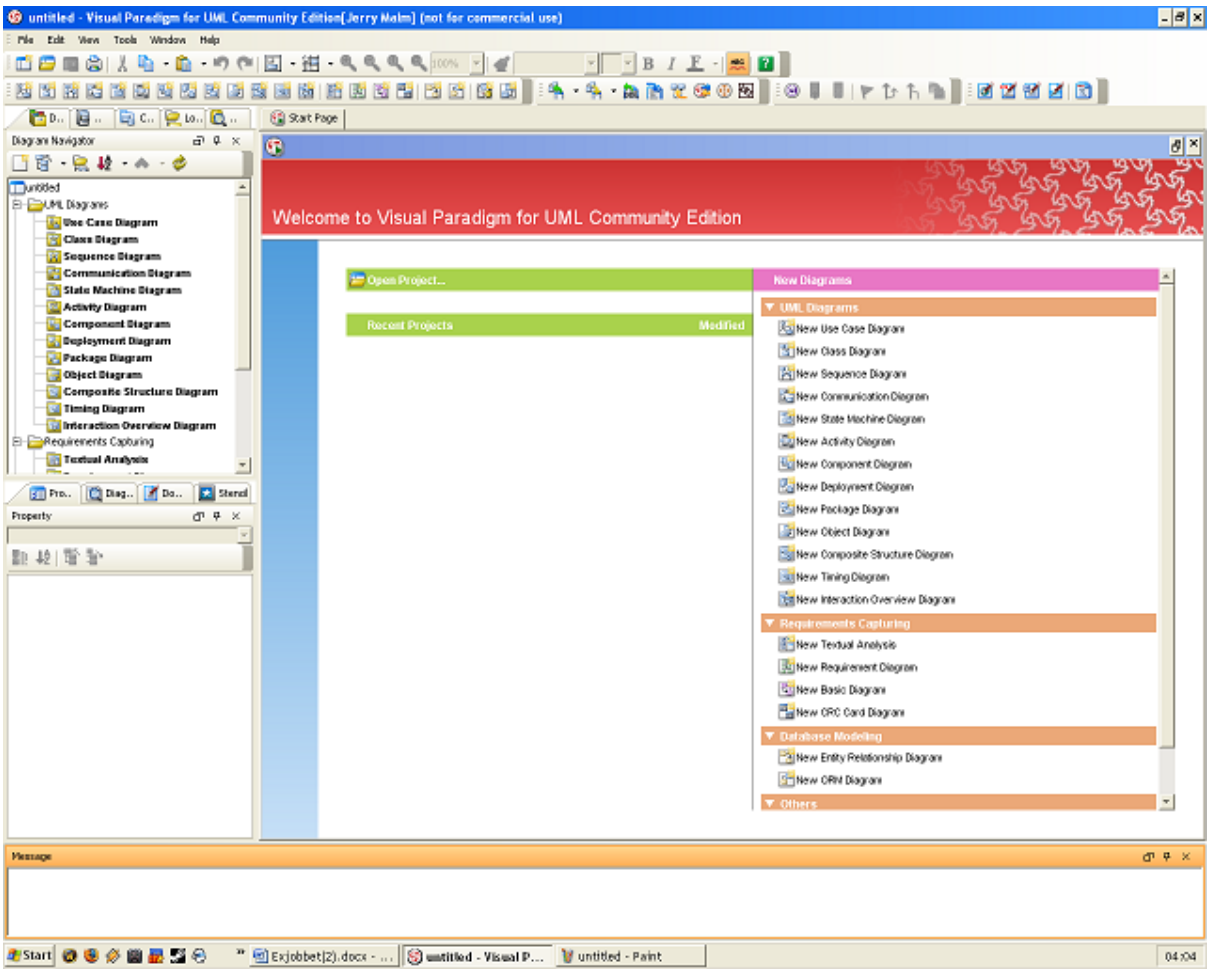
Negativt som bör undvikas:

- Ej använda rullista för att välja projekt. Ett förslag hade varit att istället visa de fem senaste öppnade projekten och samtidigt ge möjlighet att öppna vilket projekt som helst via en fildialog. Det måste dessutom finnas möjlighet att påbörja ett helt nytt projekt.
- Skapa inte ikoner som är väldigt lika varandra.
- När något nytt skapas bör vyn ändras så att det nyss skapade visas. Om användaren väljer att skapa något nytt vill hon antagligen arbeta med det direkt efter skapandet. Låt inte vyn stanna kvar vid det som arbetades med tidigare.
- Skapa inte två funktioner som i grund och botten utför exakt samma sak.
- Gör inte något onödigt svårt att utföra. Finns det en enkel lösning till att göra något är det oftast den rätta.

- Blanda inte funktioner som inte är besläktade med varandra i samma meny. Det ska också vara lätt att avgöra i vilken meny en viss funktion ska finnas. Ologiska placeringar av funktioner ska undvikas.
- Svårt att skaffa sig en uppfattning av systemet som skapades. En överblick av hur allting var länkat hade varit uppskattat.

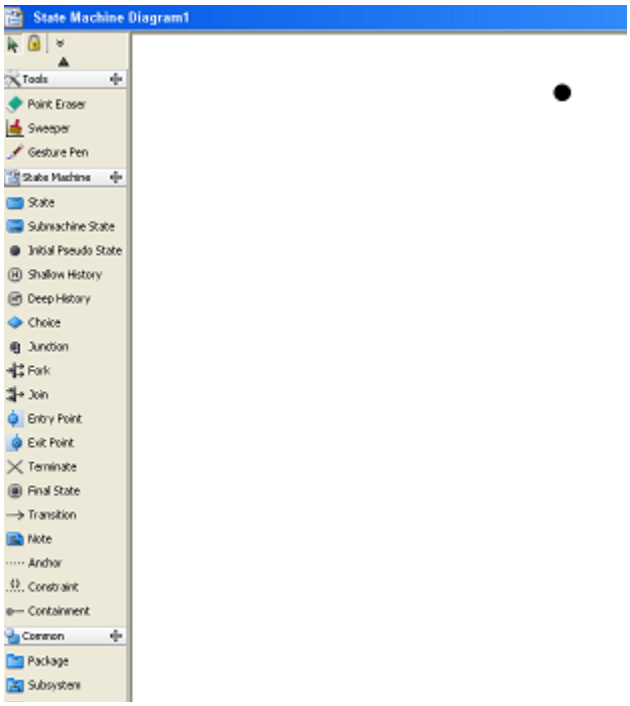
2.2.2.3 Utvärdering av Visual Paradigm for UML

I Visual Paradigm for UML går det att skapa modeller. Efter att programmet startats ser det ut enligt nedan:



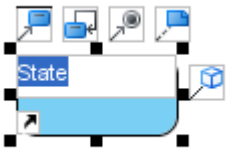
Figur 2.2.2.3.1: Startbild för Visual Paradigm for UML.

I presentationsvyn finns det massor av nyttiga funktioner. Bland annat går det att öppna ett projekt och ifall något projekt nyligen använts kommer det att visas under ”Recent Projects”. Till höger om dessa funktioner går det att skapa ett antal diagram. Eftersom vårt modellbaserade testverktyg ska använda tillståndsdiagram väljer vi ”New State Machine Diagram”. En modellyta för att rita tillståndsdiagram dyker upp och ser ut enligt nedan:



Figur 2.2.2.3.2: Modellvy för att skapa tillståndsdiagram i Visual Paradigm for UML.

När ett tillståndsdiagram skapas ritas det initiella tillstånd ut direkt. Detta är positivt eftersom det alltid måste finnas ett starttillstånd. Det verkar inte som att ett sluttillstånd måste finnas i varje tillståndsdiagram eftersom det inte skapas. Till vänster om ritytan syns verktygen för att rita ett tillståndsdiagram. Visual Paradigm har valt att använda göra ikoner som visar det som kommer att skapas samt med en beskrivande text bakom. Eftersom både en ikon och en beskrivande text ges behövs egentligen inget tooltip för knapparna men det fanns ändå. För att placera ut ett tillstånd klickar användaren först på verktyget ”State” vilket gör att verktyget markeras som valt via en blå bakgrundsfärg på knappen. Flyttas muspekaren ut till ritytan så ges ingen återkoppling förrän användaren klickar och det skapas då ett tillstånd.



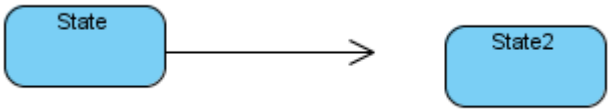
Figur 2.2.2.3.3: Ett tillstånd precis efter att det skapats i Visual Paradigm for UML.

Direkt efter att ett tillstånd skapats ges det möjlighet att editera texten som tillhör tillståndet. Andra saker som kan göras är att tillståndets storlek kan ändras om användaren drar i svarta rektanglarna. Att det finns åtta olika ställen att ändra storleken på tillståndet känns onödigt eftersom det hade räckt med fyra, en i varje hörn av tillståndet. Det finns också fem olika funktioner runtom tillståndet varav de tre första uppifrån vänster är utav intresse. Den första knappen används för att snabbt och smidigt skapa ett nytt tillstånd med en tillståndsändring ifrån det markerade tillståndet. Den andra knappen används för att göra en återkoppling till samma tillstånd och den tredje knappen används till att skapa ett sluttillstånd med en tillståndsändring ifrån det markerade tillståndet. Tyvärr visas det alldeles för många

funktioner när ett tillstånd markeras. Istället borde antalet funktioner minimeras så att användaren inte råkar trycka på en annan funktion än den önskade. För att editera texten hos ett tillstånd eller en tillståndsändring räcker det att objektet i fråga markeras och att användaren sen börjar skriva. Det ges ingen återkoppling om att det är möjligt utan det krävs att användaren dubbelklickar på ett objekt för att programmet ska visa att det är möjligt att ändra texten.

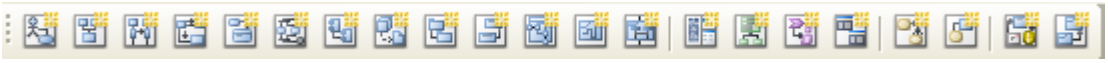
Figur 2.2.2.3.4: Återkoppling om att en text kan ändras i Visual Paradigm for UML.

Vill användaren inte använda sig utav snabbfunktionen för att länka ett tillstånd till ett befintligt går det att skapa tillståndsändringar och tillstånd med hjälp av ritverktygen på vänster sida. Tyvärr måste användaren klicka på tillståndsverktyget varje gång som hon vill sätta ut ett tillstånd eftersom verktyget avmarkeras efter att ett tillstånd ritats ut. Det hade varit bra om verktyget som var valt inte avmarkerades så att det gick att rita ut mer än ett tillstånd efter att verktyget valts. För att rita ut tillståndsändringar mellan två tillstånd måste verktyget ”Transition” väljas. Klickar användaren i ritytan utan att klicka på ett tillstånd eller dylikt händer ingenting och tillståndsändringsverktyget avmarkeras. Otroligt irriterande för användaren eftersom hon måste gå tillbaka till verktygslådan och välja om verktyget. Om användaren klickar på ett objekt skapas det en återkoppling till det objektet. För att lyckas skapa en tillståndsändring mellan två tillstånd krävs det att användaren först klickar på det tillstånd som tillståndsändringen ska utgå ifrån och sen, medan musknappen är intryckt, förflytta muspekaren till det tillstånd som tillståndsändringen ska sluta på.



Figur 2.2.2.3.5: Exempel på när en tillståndsändring mellan två tillstånd skapas i Visual Paradigm for UML.

En intressant sak är hur utvecklarna tänkte när de skapade ikonerna för de olika diagrammen som går att skapa. Det är inte direkt lätt att se någon skillnad på ikonerna utan det krävs nästan att användaren lärt sig ordning i vilken de står. Givetvis är det väldigt svårt att göra ikoner för 13 olika diagram och samtidigt göra dem så pass unika att en användare snabbt kan avgöra vilken typ av diagram som skapas. Tur som det är visas ett tooltip om användaren håller muspekaren över en av knapparna en längre tid. En bättre lösning hade varit om det funnits en knapp som när den användes visade en textbaserad rullista med de 13 olika diagramtyperna.



Figur 2.2.2.3.6: Ikonerna för att skapa nya diagram i Visual Paradigm for UML.

Positiva lärdomar att ta med sig:

- Möjlighet att öppna nyss använda projekt så att användaren slipper att leta upp det projekt som hon vill använda om hon använt det nyss.

- Rita ut starttillståndet när ett tillståndsdiagram skapas eftersom det alltid måste finnas ett sådant.
- Ikonerna för ritverktygen visar det som kommer att skapas.
- Det ritverktyg som för tillfället är valt markeras genom att bakgrundsfärgen på knappen blir blå.
- Möjlighet att ändra texten i tillståndet direkt efter att det skapats. Kan vara irriterande ifall användaren vill skapa mer än ett objekt innan hon sätter dit texten.
- Lätt tillgängliga funktioner för att snabbt länka ett nytt tillstånd till det markerade. Det gick även att göra en återkoppling till det markerade tillståndet eller att länka till ett sluttillstånd.
- Om tillståndsändringsverktyget var valt och användaren klickade på ett tillstånd skapades det en återkoppling till det tillståndet. Lätt metod för att skapa återkopplingar.
- För att skapa en tillståndsändring mellan två tillstånd klickade användaren först på det tillstånd som tillståndsändringen skulle utgå ifrån och höll sedan nere musknappen tills det avslutande tillståndet nåtts.
- Bra med tooltip till ikonknappar, särskilt för sådana knappar vars ikoner är mindre bra designade.

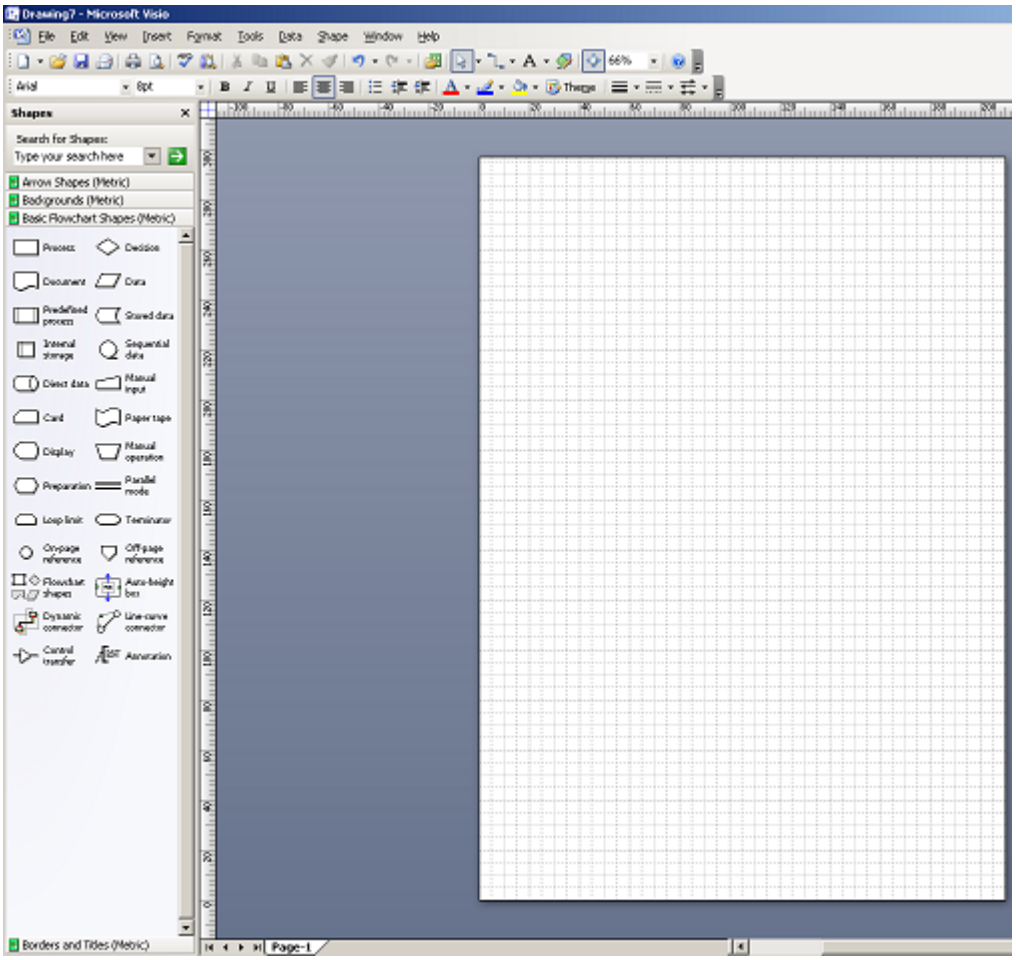
Negativt som bör undvikas:

- Om det alltid ska finnas ett sluttillstånd i ett tillståndsdiagram bör även detta ritas ut när diagrammet skapas.
- Finns det en beskrivande text till en ikon måste det inte finnas ett tooltip då det känns överflödigt.
- Ingen återkoppling gavs om exakt var ett tillstånd skulle skapas.
- Kan bli lite väl mycket detaljer hos ett tillstånd om många funktioner kan göras efter att det markerats.
- Inget verktyg var valt efter att tillståndsdiagrammet skapades. Exempelvis kunde tillståndsverktyget varit valt.
- Det gavs ingen återkoppling om att det var möjligt att ändra texten hos ett tillstånd eller en tillståndsändring om användaren bara klickade en gång. Återkopplingen gavs bara om användaren dubbelklickade på objektet.
- Valt verktyg avmarkerades efter att det använts en gång. Det borde istället fortfarande vara aktivt efter att något ritats så att det var möjligt att rita mer än ett objekt. För att rita ut flera objekt var användaren nu tvungen att välja verktyget efter varje objekt som ritades ut.
- Om användaren försökte rita en tillståndsändring utan att klicka på ett tillstånd så ritades ingenting ut alls och tillståndsändringsverktyget avmarkerades. Verktyget borde inte avmarkeras eftersom användaren aldrig fick fram någon tillståndsändring.

- Ikonerna för att skapa nya diagram var alldeles för lika varandra och gjorde det i stort sett omöjligt för en ny användare att skapa rätt diagram utan att läsa tooltipen.

2.2.2.4 Utvärdering av Microsoft Office Visio 2007

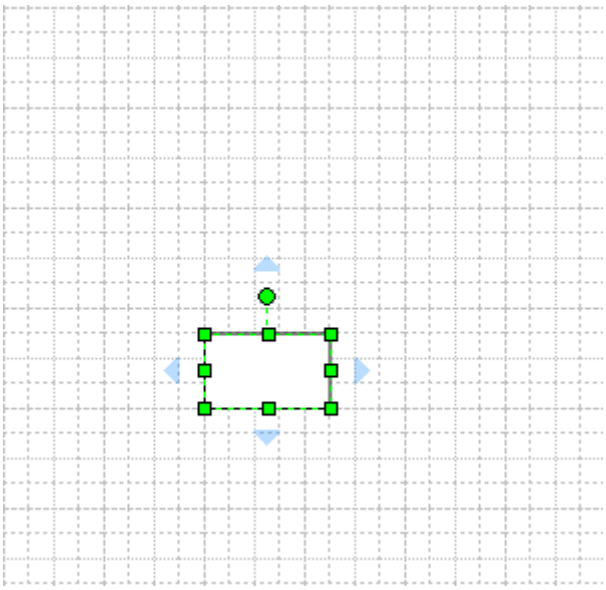
Ett program som det går att rita modeller i är Office Visio 2007 från Microsoft. Till att börja med bör det nämnas att det går att rita mer än tillståndsdiagram i detta verktyg. Dock kommer denna studie enbart att fokusera på ritandet av tillståndsdiagram. För att kunna rita tillståndsdiagram väljs först ”Flowchart” och sedan ”Basic Flowchart”. Efter det har vi en rityta som ser ut enligt figur 2.2.2.4.1 nedan:



Figur 2.2.2.4.1: Startbild för ”Basic Flowchart” i Microsoft Office Visio 2007.

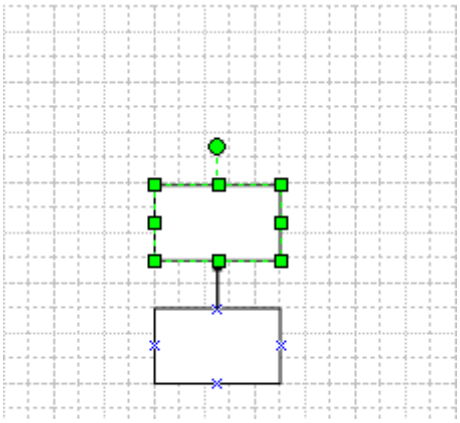
I Visio är det en tydlig uppdelning mellan ritverktygen och övrig funktionalitet. Ritverktygen är samlade på vänster sida medan övriga funktioner är placerade i toppen av programmet. Även ritytan är skild från ritverktygen vilket är positivt. Om verktygsfältet hade varit placerat i ritytan skulle det krävts att användaren flyttade verktygen för att rita på ytan precis under. Ritverktygen är uppdelade i olika flikar och för det här programmet är det bra eftersom det finns så pass många ritverktyg. Hade alla visats på en och samma gång hade det blivit för många verktyg för användaren. Något som däremot inte fungerade bra var själva fliksystemet. Det bygger på samma koncept som vi tidigare såg hos Qtronics, dvs. en del flikar ligger i övre delen av programmet och en del i nedre delen. Det blir helt enkelt för långa avstånd mellan flikarna för att det ska fungera effektivt. För att rita ut ett tillstånd trodde jag att det räckte med att markera verktyget som skulle användas och sen rita i ritytan.

Tyvärr hände det ingenting om detta gjordes utan istället var användaren tvungen att dra tillståndet från verktygslådan ut på ritytan. Efter att ett tillstånd skapats såg ritytan ut enligt nedan:



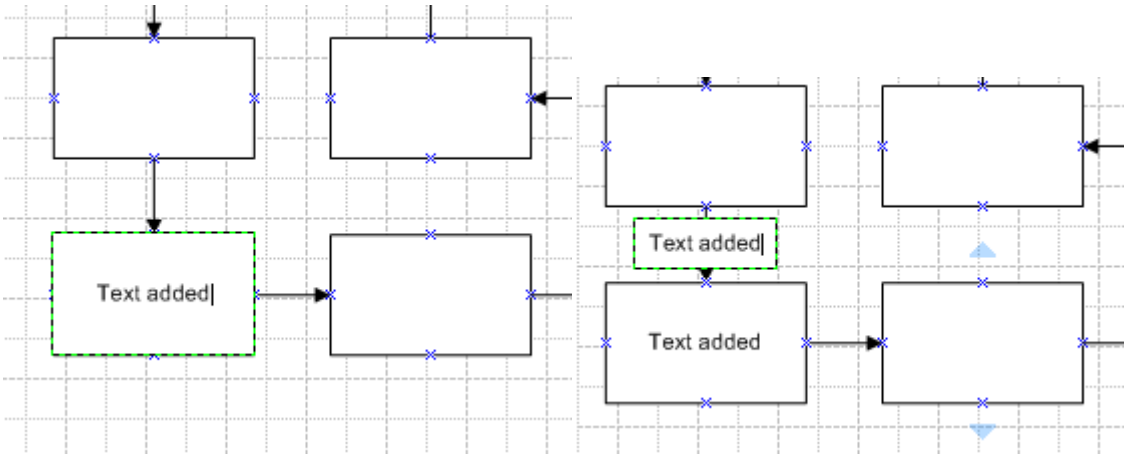
Figur 2.2.2.4.2: Bild från Microsoft Office Visio 2007 efter att ett tillstånd adderats.

Det som användaren först lägger märke till är att tillståndet har åtta gröna kvadrater längs med sin yttre gräns, en grön cirkel ovanför sig samt fyra blå pilar i varje väderstreck utanför tillståndet. Det känns som att det är lite väl många detaljer för så pass liten yta vilket kan göra det svårt för användaren att komma åt den funktion hos tillståndet som önskas. Den gröna cirkeln ovanför tillståndet låter dig rotera tillståndet. Personligen kändes det som att denna funktion inte användes särskilt mycket och därför borde den kanske inte funnits i ritytan utan istället i en meny. De åtta kvadraterna är alla olika punkter för att ändra storleken på tillståndet. Varje kvadrat ger dig möjlighet att ändra storleken på den sida där kvadraten befinner sig. Det känns lite överflödigt att ha så många som åtta olika ställen att ändra storleken på. Istället hade det räckt att det fanns möjligt att ändra storleken i de fyra hörnen eftersom det då skulle vara möjligt att ändra storleken på alla sätt och vis. De fyra pilarna, en i varje väderstreck, låter dig skapa tillstånd på ett snabbt och smidigt sätt. Ifall du klickar på den norra pilen skapas det ett tillstånd norr om det tillstånd som var markerat och en tillståndsändring skapas mellan det markerade och det skapade tillståndet. Det kan vara störigt med alla dessa extra detaljer hos tillståndet men just funktionen att lägga till tillstånd genom ett klick var en positiv funktion. För att minska detaljrikedomen hos ett tillstånd hade det varit bra om cirkeln tagits bort och om antalet rektanglar minskats till fyra.



Figur 2.2.2.4.3: Bild från Microsoft Office Visio 2007 efter att ett andra tillstånd adderats.

För att kunna lägga till text till ett tillstånd eller en tillståndsändring räcker det att markera objektet i fråga och sedan skriva texten. Det ges tyvärr ingen återkoppling om att det är möjligt att skriva text förrän användaren börjar skriva. Det går också bra att dubbelklicka på ett objekt och då syns det klart och tydligt att det går att lägga till text eftersom ett blinkande streck dyker upp i objektet.



Figur 2.2.2.4.4: Bilder från Microsoft Office Visio 2007 som visar hur det ser ut när text läggs till i ett tillstånd och sedan i en tillståndsändring.

Om en användare vill göra avancerade inställningar för ett objekt kan detta göras genom att högerklicka på ett objekt och sedan välja ”Properties”. Detta leder till ett nytt fönster där diverse inställningsmöjligheter ges. Denna lösning är bra eftersom användaren inte får för mycket information presenterad på en och samma gång. De inställningar som sällan görs göms istället i ett annat fönster. I menyn som dyker upp när det högerklickas på ett objekt finns också alternativ för att klippa ut, kopiera och klistra in vilket kan vara bra om en användare inte kan knappkombinationerna för dessa.

Om användaren inte vill använda sig av de blåa pilarna för att länka nya tillstånd till ett befintligt tillstånd går det att placera ut tillstånden precis som vanligt och sen välja att dra ut tillståndsändringar (connectors) via verktygslådan till vänster. När en tillståndsändring ritats ut kommer den först inte att vara länkad till några objekt alls utan användaren måste dra änd-

samt slutpunkten till de objekt som är start- respektive sluttillstånd. Detta är inte vidare effektivt eftersom det krävs hela tre kommandon för att skapa en tillståndsändring mellan två tillstånd. Det hade istället varit bra om det gick att placera tillståndsändringen på det tillstånd som var starttillståndet och sedan dra tillståndsändringen till det avslutandet tillståndet och på så sätt reducera antalet kommandon till två.

Positiva lärdomar att ta med sig:

- Tydlig uppdelning mellan ritverktyg och övriga verktyg. Samtidigt är ritytan tydligt skild från övriga delar av programmet.
- Genom att klicka på en detalj hos ett tillstånd efter att det markerats var det möjligt att snabbt och enkelt lägga till ett andra tillstånd som var sammankopplat med det första. Det var också möjligt att ändra storleken på ett tillstånd efter att det markerats.
- För att komma åt avancerade inställningar för ett objekt krävdes det att användaren högerklickade på ett objekt och sedan valde ”Properties” vilket ledde till att ett nytt fönster öppnades. Det är bra att inte låta användaren ha möjlighet att ändra avancerade inställningar direkt från originalfönstret eftersom det lätt kan bli för mycket information för användaren att behandla.
- Via menyn som dyker upp när en användare högerklickar på ett objekt kan användaren klippa ut eller kopiera ett objekt. Bra ifall användaren inte kan knappkombinationen för dessa funktioner.
- Att skriva text till ett objekt kunde lätt göras genom att först markera det och sedan börja skriva. Tyvärr gavs det dock ingen återkoppling om att detta var möjligt förutsatt att användaren inte dubbelklickade på objektet.
- Dela upp verktygen i olika flikar ifall det finns för många verktyg för att visa alla på samma gång. Samtidigt måste det finnas något att sortera verktygen efter annars kan det vara svårt att avgöra under vilken flik ett specifikt verktyg finns.

Negativt som bör undvikas:

- Undvik fliksystem som inte lägger flikarna direkt efter varandra. Det är inte positivt att behöva flytta muspekaren över hela skärmen för att byta flik.
- Det ska ej krävas att ett objekt dras från verktygslådan ut till ritytan. Om detta krävs betyder det att användaren måste gå till verktygslådan varenda gång hon vill lägga till ett nytt objekt och utöver detta måste hon hålla inne musknappen onödigt mycket.
- När ett objekt markerats ska det inte dyka upp onödigt många funktioner. Istället bör dessa funktioner begränsas till de som kommer användas mest. Det kan bli svårt att komma åt en specifik funktion p.g.a. mängden funktionerna som ges.
- Alldeles för omständigt att rita ut tillståndsändringar ifall detta inte gjordes via de blå pilarna på tillstånden. Det krävdes först att en tillståndsändring ritades ut på modellytan, sen var användaren tvungen att dra starten av tillståndsändringen till starttillståndet och slutet av tillståndsändringen till sluttillståndet.

2.2.3 Utvärdering av övriga program

Utvärderingen av övriga program utfördes mindre grundligt än utvärderingen av modell- och testverktygen. Denna utvärdering utfördes för att få inspiration för knappar och ikoner samt idéer för hur hur modeller skulle kunna ritas på ett effektivt och användbart sätt. Följande program testades:

- Microsoft Paint [13]
- Microsoft Office Word 2007 [14]
- Microsoft Office PowerPoint 2007 [15]
- Adobe Photoshop CS4 [16]

2.2.3.1 Microsoft Paint

Programmet Paint från Microsoft är ett program som det går att rita i med hjälp av diverse verktyg. Det som kommer att utvärderas i detta program är verktygslådan och ritytan. Verktygslådan i Paint består av 16 olika verktyg och från början är verktyget ”Penna” valt. Att just ”Penna” är valt från början kan anses som ett lämpligt val då de flesta användarna troligtvis vill börja med att använda detta verktyg. Verktyget som väljs får en ljusgrå rektangulär ram runt sig för att användaren ska kunna identifiera vilket verktyg som är aktivt, bra och tydlig återkoppling för användaren. Personligen känns det som att den ljusgrå ramen räcker men en del kanske finner det svårt att snabbt se vilket verktyg som är valt. En tydligare markering hade kanske varit att föredra, t.ex. att ramen hade fått en annan färg än ljusgrå eftersom bakgrundsfärgen för verktygslådan är snarlik. Utöver detta hade det varit önskvärt att ha någon form av avgränsning som markerar varje knapps yta då gränserna mellan knapparna för tillfället inte visas. En bild som visar hur verktygslådan ser ut vid start kan ses nedan:

Figur 2.2.3.1.1: Verktygslådan i Microsoft Paint.

Ikonerna för knapparna i Paint är till största delen hämtade från den riktiga världen. Ikonerna för verktygen penna, pensel, retuschspruta, radergummi, förstora, hämta färg och fyll är alla exempel på bra ikoner som efterliknar deras verkliga motparter. Resterande ikoner går av förklarliga skäl inte att hämta från den riktiga världen så istället visar dessa ikoner vad resultatet av att använda verktyget blir, t.ex. så går det att rita en cirkel med verktyget som har en ikon som är en cirkel. Den konceptuella modellen är välgjord eftersom användaren

ofta kan gissa sig till vad som kommer att hända. Är användaren däremot osäker på vad ett verktyg gör så går det att hålla muspekaren ovanför ikonen så visas en förklarande text, ett så kallat tooltip vilket är ett utmärkt sätt att få återkoppling på utan att irritera erfarna användare. För en del av verktygen finns det extra inställningar. Om ett verktyg som har extra inställningar väljs kommer dessa valmöjligheter att visas nedanför verktygslådan. Tack vare detta visas bara de aktuella extrainställningarna och användaren slipper att leta igenom alla verktygs extrainställningar. Ett exempel på hur det ser ut när ett verktyg med extra inställningar har valts i Paint kan ses i figur 2.2.3.1.2 nedan där verktyget ”Pensel” valts:

Figur 2.2.3.1.2: Exempel på en utökad verktygslåda i Microsoft Paint.
Verktyget som valts som ger fler valmöjligheter är ”Pensel”.

Ritytan är till att börja med en enda stor vit yta, ungefär som ett pappersark. Muspekaren som visas i ytan ändrar utseende beroende på vilket verktyg som är valt. Detta är bra återkoppling för användaren som snabbt kan avgöra vilket verktyg som är valt utan att titta på verktygslådan. Tyvärr är muspekaren samma för en del av verktygen men för de verktyg som har unika muspekare är detta en smart lösning. Något annat som också är intressant för denna rapport är var Paint väljer att rita ut saker beroende på muspekarens position samt hur användaren ska göra för att rita saker så som hon tänkt sig. Paint ritar alltid exakt där muspekaren är och beroende på muspekarens utseende är det möjligt att se hur stor del av ritytan som kommer att påverkas. För att rita en rektangel eller någon annan geometrisk figur klickar användaren en gång på den position som hon vill att rektangelns övre vänstra hörn ska starta i. Efter det håller användaren inne musknappen och drar muspekaren tills önskad storlek på rektangeln har nåtts.

Positiva lärdomar att ta med sig:

- Ikonerna ska i största möjliga utsträckning hämtas från den verkliga världen. Om det inte är möjligt ska de istället visa det som verktyget kan åstadkomma. Ikonerna ska snabbt ge användarna en uppfattning om verktygets funktion via konceptuella modeller.

- En förklarande text, så kallat tooltip, som dyker upp när muspekaren hålls över en ikon är väldigt bra återkoppling som berättar för användaren vad ett verktyg gör.
- Extra inställningar är osynliga tills ett verktyg med extra inställningar valts. Ett bra exempel på när självinstruerande använts för att begränsa informationen för användaren.
- Verktyget som är valt i början ska vara det som användaren med största sannolikhet kommer att använda sig av.
- Att använda sig av olika muspekare för olika verktygen ger användaren snabbt återkoppling om vilket verktyg som valts.

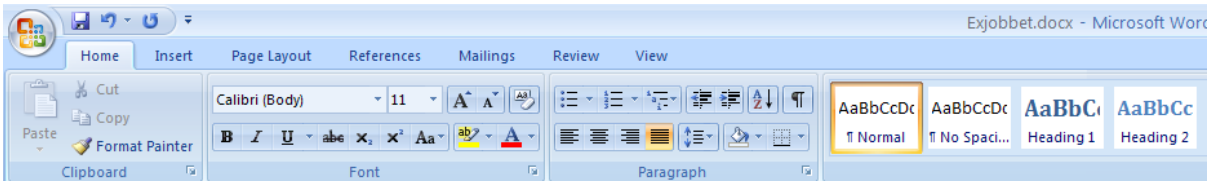
Negativt som bör undvikas:

- Inga tydliga markerade gränser mellan knapparna, kan vara svårt att se var gränserna går mellan knapparna är.
- Ej 100 % tydlig markering av vilket verktyg som valts även om markeringen i sig är positiv återkoppling. Det skulle varit önskvärt att använda sig av en färg som ej liknar den som används i bakgrunden för att tydligare markera vilket verktyg som är aktivt.

2.2.3.2 Microsoft Office Word 2007

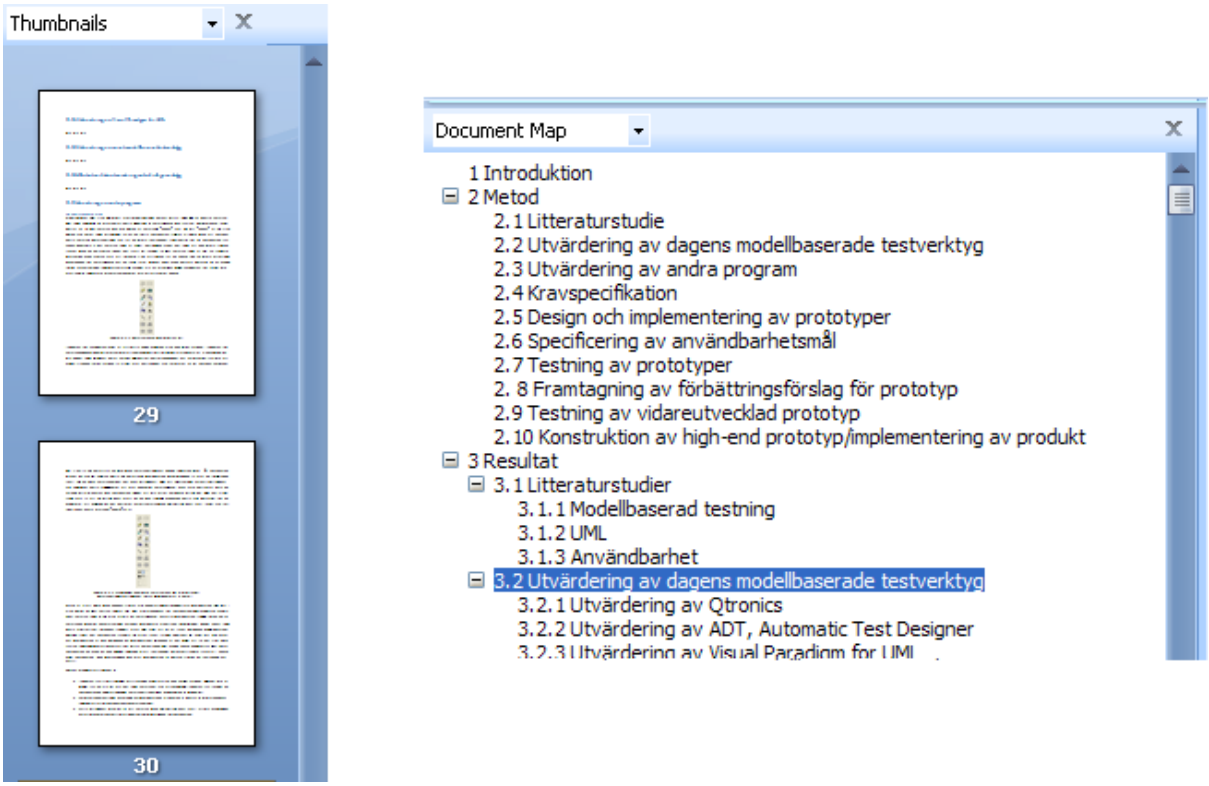
Ett annat program som också analyserades var Microsofts Word 2007 som ingår i Officepaketet. Word 2007 är ett program som det går att skapa textdokument med.

Det första som läggs märke till är något som inte påträffats tidigare i denna rapport och det är att menyerna inte följer den vanliga traditionella standarden. Istället för att endast använda sig av vanliga textmenyer med små ikoner är menyerna fyllda med stora ikoner. Funktionerna i menyerna lägger sig inte heller vertikalt utan brer ut sig horisontellt. Förutom att detta kan uppfattas som väldigt irriterande för en användare som är van vid den gamla standarden så kan det samtidigt, oavsett tidigare erfarenheter hos användaren, vara ett mindre effektivt alternativ. Detta eftersom användaren tvingas flytta musen över hela skärmen för att komma åt en funktion som befinner sig på den högra sidan. Först måste användaren välja vilken meny som ska öppnas och sen ska funktionen i menyn väljas, se bild nedan för att få en uppfattning om varför detta kan vara jobbigt. Något som däremot är positivt med de stora ikoner är att en förstagångsanvändare lätt kan förstå vad en funktion kommer att göra. Ikonerna är konstruerade på ett sådant sätt att de visar det som kommer att ske om du väljer just den funktionen. Perfekt användning av konceptuella modeller. En annan sak som också fungerade bra i menyerna var att menyalternativ som inte gick att välja vid ett visst tillfälle blev gråmarkerat och omöjligt att använda så användaren tydligt fick veta att det inte gick att använda. Ett exempel på utmärkt användning av restriktioner är t.ex. att om en bild markeras så går det inte längre att byta typsnitt.



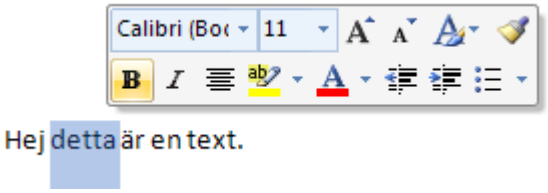
Figur 2.2.3.2.1: Menyn ”Home” i Microsoft Office Word 2007.

På vänster sida om skrivytan finns det en dokumentöversikt som antingen visar en förhandsvisning av sidorna eller en överblick av alla kapitel som skrivits. Det är enkelt att navigera mellan sidor eftersom det för varje sida visas en liten miniatyrversion. Alltså är det möjligt att få en uppfattning om hur sidan ser ut innan den väljs. Navigeras det mellan kapitel så visas deras namn och position i dokumentet vilket gör att det snabbt och lätt går att ta sig till rätt avsnitt. För att skifta mellan översikterna används en rullista. I bilden nedan ses förhandsvisning av sidorna till vänster och en överblick av alla kapitel till höger.



Figur 2.2.3.2.2: Dokumentöversikt för att snabbt ta sig till ett specifikt kapitel eller en specifik sida.

En funktion som fungerade bra i Word 2007 var en meny som dök upp om en användare markerade ett ord eller textstycke. Först är menyn transparent men om muspekaren flyttas till den så kommer den att synas tydligt. I denna meny finns de funktioner som en användare vanligtvis använder sig av, t.ex. att göra en text fetstilt eller kursiv. Denna funktion är bra eftersom om ett textstycke markeras så vill användaren med största sannolikhet ändra typsnitt eller något liknande. Ett problem som kan uppstå med denna meny är ifall användaren råkar få fram menyn fastän att hon inte vill använda sig av den. Det är viktigt att menyn endast kommer fram när användaren verkligen vill det.



Figur 2.2.3.2.3: Meny med de vanligtvis mest använda funktionerna som dyker upp efter att en text markerats i Microsoft Word 2007.

Positiva lärdomar att ta med sig:

- Miniatyrer eller strukturerad uppdelning av olika avsnitt eller sidor gör det lätt att navigera mellan dem. Miniatyrer ger användaren snabb återkoppling om vad som kommer att visas och samma ska gäller för den strukturerade uppdelningen.
- En snabbmeny med de vanligtvis mest använda funktionerna är en bra idé om en användare har svårt för att lära sig knappkombinationer som t.ex. att spara är ”Ctrl+S”.
- Ikoner som visar vad en funktion gör är effektiva eftersom de ger användaren en uppfattning om vad som kommer att hända om hon väljer funktionen.
- Att gråmarkera, inaktivera, funktioner som ej går att användas för tillfället är en smart lösning för att förbjuda användaren att använda ”förbjudna” funktioner.

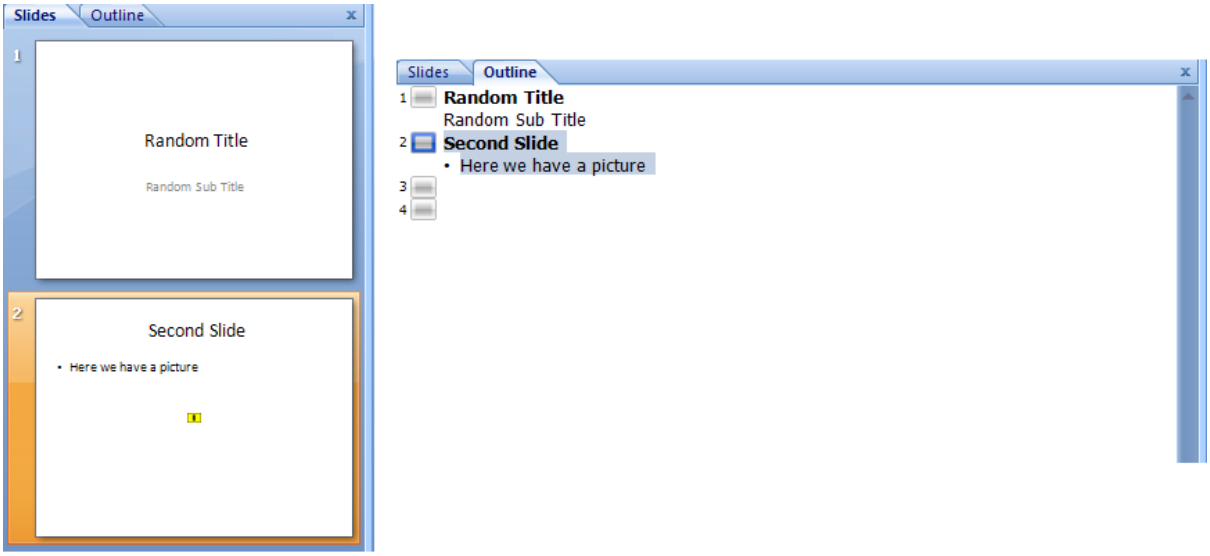
Negativt som bör undvikas:

- Snabbmenyn som dyker upp kan vara irriterande för användaren om hon av misstag råkar ta fram den. Viktigt att den endast dyker upp när användaren verkligen vill använda den.
- Försök att undvika långa sträckor mellan menyer och funktioner som hör ihop. Det är ineffektivt för användaren att behöva förflytta muspekaren över hela skärmen för att komma åt den funktion som önskas.

2.2.3.3 Microsoft Office PowerPoint 2007

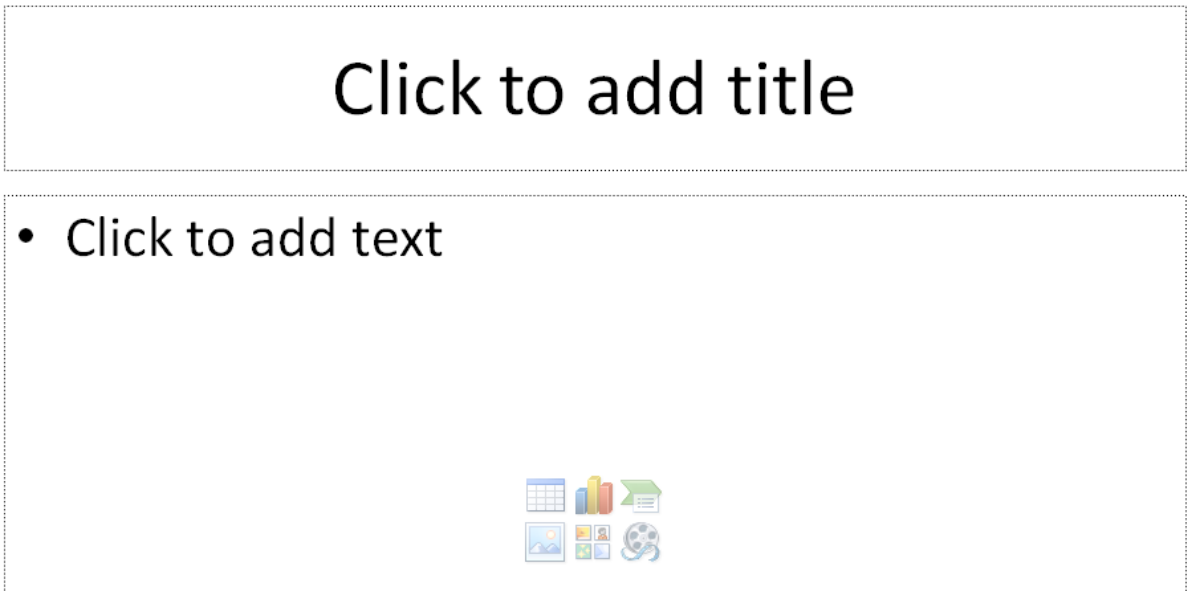
Office PowerPoint 2007 är ett program från Microsoft som det går att skapa presentationer i. Varje presentation består av en till flera slides som i programmet representeras av ett vitt pappersark. Många av funktionerna fungerar på samma sätt som i Word 2007 och därför kommer endast de funktioner som inte är identiska att presenteras.

På vänster sida om redigeringsytan finns det en översikt över alla slides som ingår i presentationen. I översikten har varje slide en miniatyr av hur sliden ser ut. Detta gör det lätt för användaren att avgöra vilken av slidesen som hon vill redigera eller titta på. I princip är det samma översikt som användes i Word 2007 fast med en skillnad och det är metoden för att skifta mellan att visa miniatyrer eller att visa slidesen i textform. I Word 2007 var det en rullista som användes men i PowerPoint 2007 är det istället flikar som används. Personligen känns det som att flikar är att föredra eftersom en rullista inte visar vad det finns för alternativ till det nuvarande valet. Hur det hela ser ut kan beskådas i figur 2.2.3.3.1 nedan:



Figur 2.2.3.3.1: Presentationsöversikt i Microsoft Office PowerPoint 2007. Vänstra delen visar miniatyrer av slidesen och den högra delen visar slidesen i textform.

Det som skiljer sig mest från Word 2007 är sättet som en presentation skapas på jämfört med hur ett dokument skapades. PowerPoint 2007 har två fördefinierade fält att redigera i för varje slide. I det övre av dessa är det endast möjligt att redigera text, närmare bestämt rubriken, och i det nedre fältet kan text, bilder och mycket annat läggas till. I figur 2.3.3.2 kan sex olika ikoner ses i det nedre fältet. Dessa är snabbgenvägar för att t.ex. lägga till tabell, diagram, bild, film eller ljud. Det är positivt att det finns knappar för dessa funktioner då de används frekvent. Tyvärr är återkopplingen lite mindre bra eftersom en användare kan tro att knapparna kommer att synas på slidesen trots att de är halvt genomskinliga. Dessa knappar försvinner så fort en användare redigerar det nedre fältet och detta kan ses som både något positivt och något negativt. Det som är positivt är att användaren inte längre kommer tro att knapparna kommer att synas på sliden. Det funkar däremot mindre bra om användaren först skriver lite text och sen bestämmer sig för att lägga till en bild då hon tvingas att använda sig av menyn. Övrigt som bör nämnas är att ikonerna för knapparna ger användaren bra information genom konceptuella modeller, har en knapp en tabell som ikon kommer det med största sannolikhet att läggas till en tabell.



Figur 2.2.3.3.2: Slidevyn i Microsoft Office PowerPoint 2007.

Positiva lärdomar att ta med sig:

- För att skifta mellan olika vyer är flikar att föredra framför en rullista eftersom en rullista gömmer alla alternativ utom det nuvarande. Vill användaren byta vy till textformat är det inte lika lätt att göra det i Word 2007 som i PowerPoint 2007.
- Lägg till knappar för de mest använda funktionerna så att användaren slipper att använda sig av menyn.

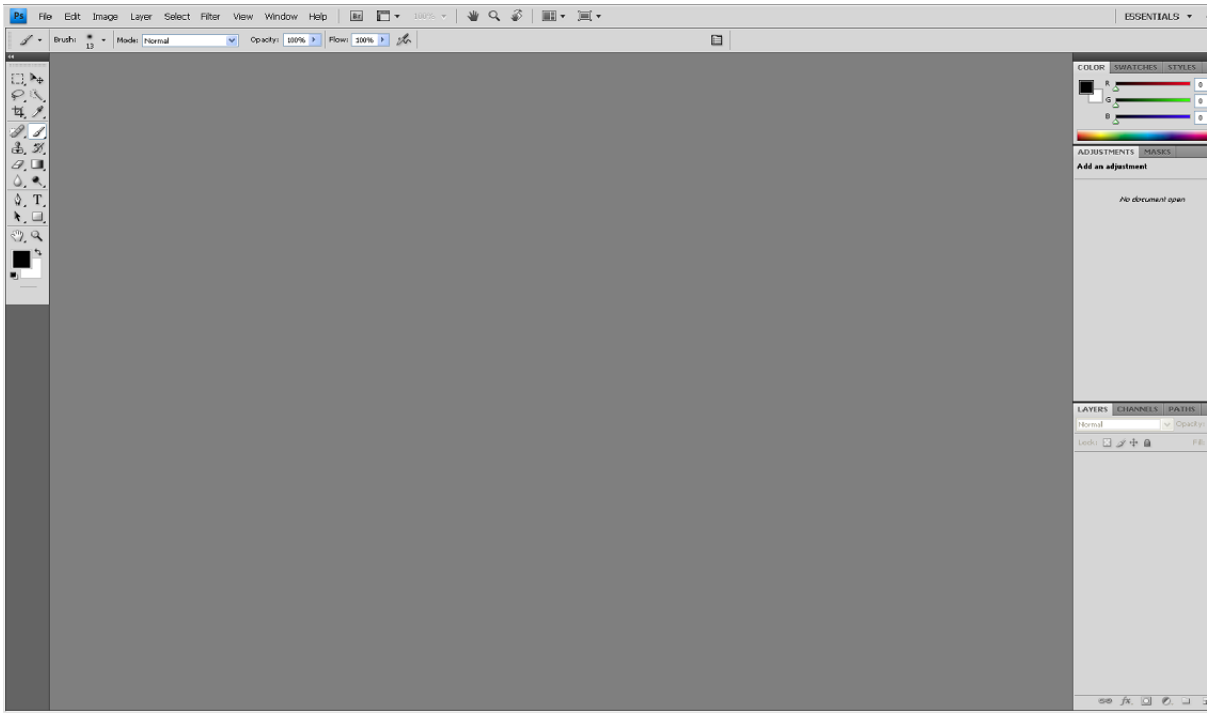
Negativt som bör undvikas:

- Undvik att lägga knappar eller liknande i redigeringsytan. Användaren kan irriteras av knapparnas positionering om den hamnar i vägen för det område som ska redigeras. Samtidigt kan det ge felaktig återkoppling då användaren kan tro att dessa knappar kommer att finnas i det sparade eller utskrivna materialet.
- Knapparna för de mest använda funktionerna ska inte försvinna efter att en av funktionerna använts.

2.2.3.4 Adobe Photoshop CS4

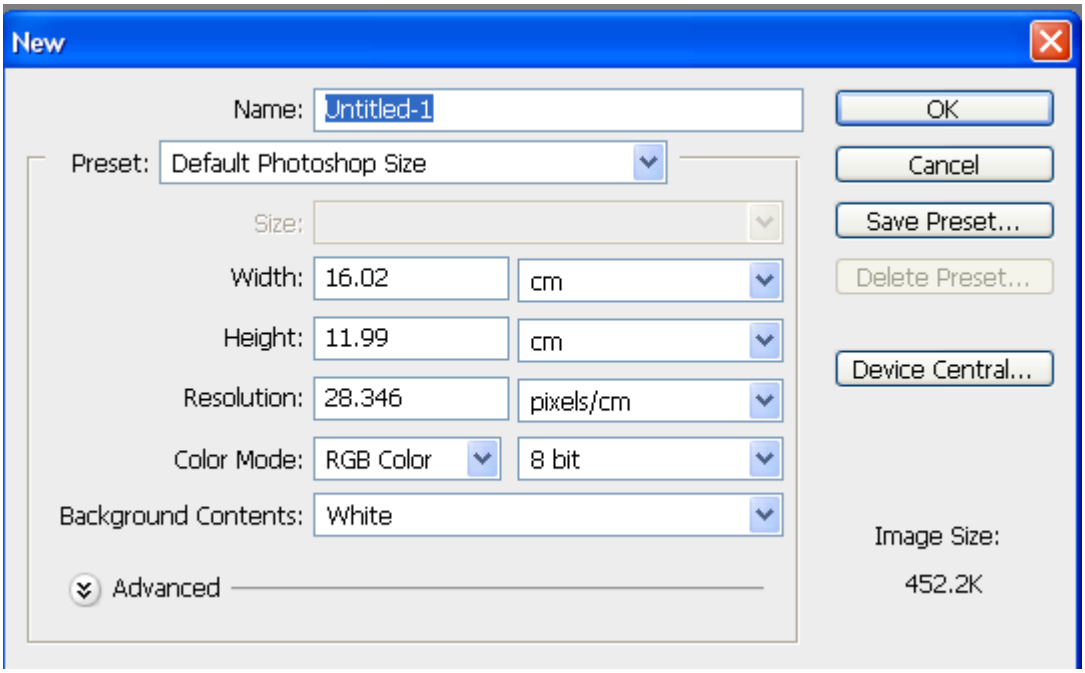
Ett mer avancerat ritprogram än Microsoft Paint är Photoshop CS4 från Adobe. Efter att programmet har startats ser programmet ut som i figur 2.2.3.4.1 nedan. I det vänstra hörnet av programmet kan verktygslådan med alla ritverktyg hittas. Alla ikonerna för ritverktygen är gjorda i svart-vitt och det hade kanske varit bättre om de istället varit i färg eftersom det ger användaren mer information. Idén bakom ikonerna i Photoshop är att de antingen visar det de ska göra, exempelvis markeringsverktyget, eller att de ska efterlikna verktyget från den verkliga världen, exempelvis verktyget för att rita har ikon med en pensel. Tyvärr är det trots detta inte alltid lätt att se vad en knapps funktion är och då hade ett tooltip varit till stor hjälp. I Photoshop finns det inga tooltip överhuvudtaget. Efter att programmet startats är verktyget för att markera ett område i ritytan vald. Detta markeras genom att verktyget får en svart

rektangel runt sig och bakgrundsfärgen hos verktyget blir vitt. Att markeringsverktyget är valt från början är ett konstigt val om en ny rityta skapas men kanske ett vettigt val om en bild öppnas.



Figur 2.2.3.4.1: Startbild från Adobe Photoshop CS4.

För att få fram en yta att rita i måste användaren gå via menyn ”File” och välja ”New”. Efter att detta gjorts dyker ett nytt fönster upp där diverse inställningar kan göras innan en rityta skapas, t.ex. kan storleken på ritytan anges i allt från centimeter till pixlar. Exakt vad som går att ställa in går att se i figur 2.2.3.4.2. Enligt mig borde det skapas en rityta automatiskt vid starten av programmet, inställningarna skulle kunna ha standardvärden som angetts av utvecklarna. Ett annat alternativ hade varit att låta användaren ha möjlighet att ange värden som sen används till den rityta som skapas automatiskt vid starten.



Figur 2.2.3.4.2: Inställningar som kan göras innan en ny rityta skapas i Adobe Photoshop CS4.

För varje verktyg finns det extra inställningar som kan göras. Dessa visas under menyn och ändras beroende på vilket verktyg som är valt för tillfället. De inställningar som kan göras för verktyget pensel visas i figur 2.2.3.4.3 nedan.

Figur 2.2.3.4.3: Extra inställningar som kan göras för verktyget pensel i Adobe Photoshop CS4.

Till höger om menyn finns det knappar för att zooma och flytta fokuset i bildytan. Ikonerna för dessa knappar är även de välgjorda eftersom de föreställer något ifrån den verkliga världen. Det är också bra att funktionaliteten är samlad på ett och samma ställe. Vill användaren zooma är det ganska troligt att hon också vill flytta sig till en specifik del av bilden efter att det zoomats.

Figur 2.2.3.4.4: Funktioner för att zooma eller flytta en bild i Adobe Photoshop CS4.

Muspekaren i Photoshop fungerar i likadant som i Microsoft Paint. Beroende på vilket ritverktyg som är valt för tillfället så anpassas muspekaren efter det. Användaren får då snabb återkoppling om vilket verktyg som för tillfället är valt och formen på muspekaren instruerar användaren hur det som ritas kommer att se ut. Det går, till skillnad från Paint, att ha mer än en rityta öppen på en och samma gång. För att hantera detta använder Photoshop sig utav flikar som det går att växla mellan. Den som är vald för tillfället får en mörkare grå bakgrund medan de som inte visas för tillfället har en ljusare nyans av grå.

Figur 2.2.3.4.5: Flikarna för att skifta mellan olika bilder i Adobe Photoshop CS4.

Positiva lärdomar att ta med sig:

- Sätt det ritverktyg som en användare med största sannolikhet kommer att börja använda som valt verktyg vid start.
- Det verktyg som väljs bör markeras tydligt på något sätt, t.ex. genom att bakgrundsfärgen för verktyget ändras till en annan färg.
- Ikonerna för knapparna hämtar inspiration antingen från vad de ska utföra eller från saker från den riktiga världen.
- Extra inställningar för ett verktyg visas bara om det är valt.
- Fliksystem smidigt för att växla mellan olika ritytor.
- Muspekaren anpassas efter vilket ritverktyg som för tillfället är valt. Detta gör att användaren snabbt kan avgöra vilket verktyg som är aktivt. En annan positiv sak med muspekarna var att de också gav användaren en liten föraning om hur det som ritades skulle se ut.

Negativt som bör undvikas:

- Mindre positivt att ingen rityta skapas direkt efter start. Det är ändå något som de flesta användarna vill ha.
- Inga tooltip för knapparna. Hade varit bra för användare som aldrig använt programmet tidigare.
- Ikonerna var gjorda i svartvitt. Det hade varit möjligt att ge användarna mer återkoppling och information ifall ikonerna istället varit i färg.

2.2.4 Slutsatser för gränssnittsstandard för befintliga modell- och testverktyg

De viktigaste slutsatserna som drogs sammanfattas i detta avsnitt. En designer ska sträva efter att:

- Visa vad som är satt och vad som måste sättas för att användaren ska kunna starta en testgenerering eller exekvering.
- Använda grå färg för att indikera att inaktiva knappar och menyalternativ inte går att använda.
- Använda ikoner som hämtar information ifrån den riktiga världen eller som försöker efterlikna funktionaliteten.
- Göra tydliga avgränsningar mellan funktionalitet som inte hör samman.
- Använda knappkombinationer så att avancerade användare kan öka sin effektivitet.
- Ge användaren återkoppling om vad som händer beroende på de val som görs.

- Använda kryssrutor där något kan antingen väljas eller väljas bort.
- Använda progressbars för att visa hur ett förlopp fortlöper.
- Visa tiden det tar att generera eller exekvera testfall.
- Använda färgkodning för synliggöra för användaren vad som är bra eller dåligt.
- Ge användaren möjlighet att öppna nyss använda projekt.
- Rita ut starttillståndet direkt när ett tillståndsdiagram skapas.
- Använda tydlig markering för vilket verktyg som för tillfället är valt.
- Ge möjlighet att använda snabbkommandon för att länka nya tillstånd från ett markerat tillstånd.
- Låta användaren skapa en tillståndsändring genom att dra ifrån starttillståndet till det avslutandet tillståndet.
- Ej kräva att användaren ska välja verktyg direkt efter att ett objekt skapats.
- Använda tooltip för ikonknappar.
- Sätta ett verktyg som valt när ett nytt tillståndsdiagram skapas.
- Sortera ritverktygen på ett ställe och övrig funktionalitet på ett annat.
- Sortera ritverktyg i olika flikar där liknande ritverktyg sorteras tillsammans om antalet verktyg är många.
- Visa inte avancerade inställningar tillsammans med vanliga inställningar. Givetvis ska det ges möjlighet att ändra även avancerade inställningar men de kan gömmas i ett annat fönster.
- Alltid fråga användaren innan något raderas.
- Endast visa extrainställningar för det verktyg som är valt för tillfället. Onödigt att visa extrainställningar för alla verktygen på en och samma gång.
- Använda flikar för att växla mellan olika filer eller vyer
- Presentera varje sida eller modell som en miniatyr. Detta gör att användaren lätt kan avgöra vilken av dem som hon vill arbeta med.
- Ändra muspekarens utseende beroende på vilket verktyg som är valt. Ge användaren återkoppling så att hon kan avgöra vilket som är aktivt och hur det som ska ritas kommer att se ut.
- Skapa tydliga gränser mellan knappar så att användaren vet exakt vilket verktyg som kommer att väljas när hon klickar på ett specifikt ställe.

Undvik att:

- Skicka fel signaler till användaren genom att använda färgkodning på ett mindre bra sätt. T.ex. ska inte en startknapp vara grön ifall den inte går att använda.
- Använda tooltip ifall en beskrivande text redan ges i knappen tillsammans med ikonen.
- Inte ge användaren någon återkoppling om var ett objekt kommer att skapas.

- Placera ut saker som ska göras i en onaturlig ordning då användaren lätt kan missa något då.
- Använda knappkombinationer som redan är allmänt accepterade för någon annan funktionalitet.
- Designa ikoner som är lika varandra.
- Placera ut alldeles för många snabbkommandon och funktioner runt ett objekt efter att det markerats.
- Sakna knappar för funktioner som används väldigt mycket. Användaren ska inte behöva gå via menyn varje gång.
- Använda rattar.
- Använda dåligt designade fliksystem som kräver att användaren flyttar musen över hela skärmen.
- Dölja sådant som går att göras. T.ex. gavs det ingen återkoppling om att det var möjligt att ändra texten hos ett tillstånd eller en tillståndsändring efter att det markerats.
- Tvinga användaren att dra ut objekt från ritverktygen.
- Vyn inte bytas till det diagram eller liknande som skapas eftersom användaren med största sannolikhet vill arbeta med det som hon skapade.
- Skapa två funktioner som i grund och botten utför exakt samma sak.
- Placera alla menyalternativ i samma meny. Sortera istället dem i rätt menyer så att användaren lätt kan hitta det hon vill använda.
- Placera knappar och liknande i redigeringsytor. Detta för att inte irritera användaren när hon arbetar i ytan, t.ex. att de kommer i vägen för det som ska göras.
- Använda svartvita ikoner utan använd istället ikoner i färg eftersom dessa kan ge mer information än de svartvita.
- Tvinga användaren till att skapa något som ändå önskas vid starten av programmet. Startas t.ex. Photoshop, utan att användaren startar det via en bild, bör en rityta skapas vid starten.

2.3 Kravspecifikation

Innan skisserna för prototyperna kunde tas fram skrevs en kravspecifikation. Dessa krav togs fram vid ett möte på System Verification tillsammans med Bobby Kociski och Johan Tejle. Anledningen till att detta gjordes var för att kunna identifiera all funktionalitet som användargränssnittet skulle ha. Utan kravspecifikationen skulle säkerligen ett antal funktioner ha saknats i prototyperna. De tre övergripande funktionerna hos programmet är:

1. Det skall vara möjligt att skapa och editera tillståndsdiagram.
2. Det skall vara möjligt att generera testfall utifrån en testfallsbeskrivning och tillståndsdiagram.
3. Det skall vara möjligt att exekvera tester utifrån en testsvit.

En fullständig lista över de funktionella kraven för användargränssnittet finns nedan. Listan utökades något efter ett andra möte hos System Verification eftersom en del funktionalitet hade missats.

2.3.1 Tillståndsdiagram

R1.1. Det skall vara möjligt att skapa ett nytt tillståndsdiagram.

R1.2. Det skall vara möjligt att editera ett tillståndsdiagram.

R1.3. Det skall vara möjligt att importera ett tillståndsdiagram till en modell.

R1.4. Det skall vara möjligt att exportera tillståndsdiagram till en fil.

2.3.2 Modell

R2.1. Det skall vara möjligt att skapa en ny modell.

R2.2. Det skall vara möjligt att ladda en sparad modell.

R2.3. Det skall vara möjligt att spara en modell.

R2.4. Det skall vara möjligt att komma åt ett tillståndsdiagram utifrån en modell.

2.3.3 Testgenerering

R3.1. Det skall vara möjligt att generera testfall utifrån en specificerad testfallsbeskrivning och tillståndsdiagram.

R3.2. Det skall vara möjligt att välja till vilken plattform testerna ska genereras, t.ex. Java eller .NET.

R3.3. Det skall vara möjligt att ange testalgoritm för ett specifikt tillståndsdiagram.

R3.4. Det skall vara möjligt att ange testalgoritm för en modell.

R3.5. Det skall vara möjligt att ange vilken kombination av statistisk data som skall tolkas som stoppkriteria för testfallsgenerering.

R3.6. Det skall vara möjligt att följa en testgenerering på ett sådant sätt att användaren får tillräckligt med återkoppling för att förstå vad som händer.

R3.7. Det skall vara möjligt att starta en testgenerering.

R3.8. Det skall vara möjligt att pausa en testgenerering.

R3.9. Det skall vara möjligt att stoppa en testgenerering.

R3.10. Det skall vara möjligt att spara en serie av testfall som genererats. Dessa testfall sparas som en testsvit.

R3.11. Det skall vara möjligt att se vilka tillstånd och tillståndsändringar som täcks av testfallen som genereras.

2.3.4 Testexekvering

R4.1. Det skall vara möjligt att öppna en testsvit för exekvering.

R4.2. Det skall vara möjligt att exekvera enstaka testfall ifrån en testsvit.

R4.3. Det skall vara möjligt att välja alla testfallen i en testsvit.

R4.4. Det skall vara möjligt att välja bort alla testfallen i en testsvit.

R4.5. Det skall vara möjligt att starta en testexekvering.

R4.6. Det skall vara möjligt att pausa en testexekvering.

R4.7. Det skall vara möjligt att stoppa en testexekvering.

R4.8. Det skall vara möjligt att se vilka tillstånd som testas under en testexekvering.

R4.9. Det skall vara möjligt att spara teststatistik från en individuell testexekvering.

R4.10. Det skall vara möjligt att spara teststatistik från en serie av testexekveringar.

R4.11. Det skall vara möjligt att kolla teststatistisk från en individuell testexekvering.

R4.12. Det skall vara möjligt att kolla teststatistik från den senaste testexekveringen.

R4.13. Det skall vara möjligt att kolla teststatistik från alla testexekveringar.

R4.14. Det skall vara möjligt att ladda sparad teststatistik.

2.3.5 Statistik och rapport

R5.1. Det skall vara möjligt att skriva ut en testrapport ifrån programmet.

R5.2. Det skall vara möjligt att generera en testrapport till en fil, t.ex. i pdf-format.

2.3.6 Övriga krav

R6.1. Programmet skall ha en ikon.

R6.2. Det skall vara möjligt att avsluta programmet via ikon och meny.

R6.3. Det skall vara möjligt att minimera programmet.

R6.4. Det skall vara möjligt att ändra storlek på programfönstret.

R6.5. Det skall finnas en hjälpmeny.

R6.6. Det skall vara möjligt att söka på ett specifikt ord i hjälpmenyn.

R6.7. Det skall vara möjligt att använda knappkombinationer, t.ex. ”Ctrl+S” för att spara, för att snabbt komma åt en önskad funktion.

2.4 Design och implementering av prototyper

Med hjälp av de lärdomar och kunskaper som förvärvades under de fyra föregående arbetsmomenten kunde två enkla skisser för hur ett användargränssnitt för modellbaserad testning skulle kunna se ut tas fram. Designerna skissades på papper för att senare implementeras i Java [17]. Genom att först rita dem på papper sparades det in en massa tid eftersom det tidigt kunde avgöras om en idé var bra eller inte. Det hade tagit längre tid att kontrollera detta i Java då det tar betydligt längre tid att koda ett användargränssnitt än att skissa det på papper.

Programmeringsspråket Java valdes dels på grund av en längre tids erfarenhet av språket och dels eftersom det är relativt lätt att utveckla gränssnitt i. Andra alternativ som fanns i åtanke i början av examensarbetet var programmeringsspråket Ruby [18] och Microsoft Office PowerPoint. Av de tre alternativen valdes Microsoft Office PowerPoint bort först. PowerPoint kan lättast jämföras med att producera prototyper i pappersform. Varje sida i en PowerPoint-presentation motsvarar ett pappersark som visar en specifik del av användargränssnittet vid ett specifikt tillfälle. Fördelarna gentemot att producera prototyperna i pappersform är att försökspersonen interagerar med prototypen med mus och tangentbord. Nackdelarna med att göra prototyperna i PowerPoint var att det skulle krävas väldigt mycket arbete för att producera alla sidorna och sen skulle dessutom sidorna behöva länkas till varandra. Största bekymret skulle ändå vara att om något ändrades på en sida så skulle det behöva ändras på alla andra också. Om något istället ändrades i Java hade det reflekterats på alla sidorna utan att ändringen behövde göras på flera ställen. Samtidigt är det inte lätt att göra en del funktioner i pappersform. Det hade t.ex. varit väldigt svårt att simulera ritandet av modeller i PowerPoint. På grund av dessa uppenbara brister valdes PowerPoint bort till fördel för ett vanligt programmeringsspråk. Det tredje och sista alternative var Ruby. Ruby är ett programmeringsspråk som är lätt att programmera i p.g.a. en enkel och naturlig syntax. En annan fördel med Ruby är att det har stöd för felhantering, vilket gör det enkelt att hantera fel som uppstår . Anledningen till att Java valdes framför Ruby var att det inte fanns någon direkt fördel med att använda Ruby samt det faktum att författaren hade en längre tids erfarenhet av utveckling i Java. Anledningen till att det bestämdes att prototyperna skulle tillverkas med hjälp av ett programmeringsspråk var att komma så nära det verkliga användargränssnittet som möjligt, dvs. att prototyperna skulle gå att testa på en dator och styras med hjälp av mus och tangentbord. Det skulle varit möjligt att testa prototyperna i pappersform men det hade varit långt från optimalt. För det första hade försökspersonerna inte interagerat med prototyperna på det sätt som de skulle gjort med det riktiga användargränssnittet och för det andra hade det varit väldigt jobbigt att byta mellan en massa pappersark för varje val som gjordes.

Efter att det stod klart att prototyperna skulle utvecklas i Java var nästa val att välja en utvecklingsmiljö. Valet låg först på Eclipse [19] eftersom författaren tidigare använt sig av detta verktyg både i skolan, på arbetsplatser samt på sin egen fritid. En av fördelarna med att arbeta i Eclipse är att användaren inte behöver skriva ut ett helt metodanrop. Har ett metodanrop börjat skrivas kommer det upp alternativ beroende på vilka bokstäver som skrivits. Genom att trycka *Enter* slipper programmeraren skriva hela metodnamnet och sparar på så sätt tid. Samtidigt som alternativen visas så skrivs det även ut information om metoden som är markerad vilket gör att programmeraren slipper att läsa dokumentation på Internet. När alternativa utvecklingsmiljöer till Eclipse söktes kom författaren i kontakt med NetBeans [20]. NetBeans har förutom de båda fördelarna som nämndes ovan för Eclipse även stöd för att rita upp användargränssnitt utan att någon kod skrivs. Det går till på ett sådant sätt att

användaren först väljer en komponent som ska läggas till i användargränssnittet och sen placeras ut den. Det som visas på skärmen när användargränssnittet ritas upp är det som sen fås när programmet körs. Förutom möjligheten att placera ut en komponent exakt där det önskas går det även att ändra storleken på komponenterna genom att dra i kanterna. Detta gör att mycket tid sparas in på att sitta och ändra en komponents storlek med ett fåtal pixlar. Relativt tidigt efter att ha testat NetBeans bestämdes det att Eclipse valdes bort till förmån för NetBeans. Den uppenbara fördelen med att slippa ändra i koden och kompilera om bara för att få en knapp fem pixlar breddare var den avgörande faktorn. Tyvärr fanns det en nackdel med detta och det var att det var arbetsamt att ändra i den kod som genererats av NetBeans. Lyckligtvis behövdes detta inte göras alldeles för många gånger. Tack vare att en massa tid sparades in på att rita upp användargränssnittet kunde mer energi läggas på att implementera den underliggande funktionaliteten.

Det sista som skulle bestämmas var vilken arbetsmetodik som skulle följas. Inga direkta efterforskningar gjordes här utan valet föll snabbt på XP, Extreme Programming [21]. I XP utförs arbetet i flera små iterationer istället för få stora som t.ex. är fallet med vattenfallsmodellen [22]. Fördelarna med att arbeta i flera små iterationer är att det snabbt går att testa funktionalitet som implementerats och att det snabbt och lätt går att ändra något som implementerats men som helt plötsligt ska vara på ett annat sätt. Eftersom arbetet görs i flera små iterationer kan det kontinuerligt släppas nya releaser som testas för att hitta eventuella fel vilket gör att fel åtgärdas så snabbt som de upptäcks. Arbetet för varje iteration bestäms via så kallade *user stories* [23]. En *user story* beskriver en deluppgift som ska göras samt inkluderar en uppskattning av hur lång tid det kommer ta att genomföra. För varje iteration väljs en eller flera *user stories* ut som ska implementeras innan nästa iteration påbörjas. På grund av detta är arbetet alltid uppdelat i mindre, detaljerade uppgifter så att stora, odetaljerade beskrivningar undveks. Samtidigt väljs *user stories* ut så att utvecklaren får lagom mycket arbete att göra varje vecka. En sista sak som bör nämnas om XP är att målet först och främst är att få något till att fungera. När något väl fungerar kan det i efterhand optimeras genom att refaktorisera den kod som skrivits.

När det väl blev dags att börja designa prototyperna bestämdes det att två prototyper skulle produceras. Dessa två prototyper skulle ha skilda grunddesigner för att på ett enkelt sätt analysera vilken av grunddesignerna som var att föredra vid vidareutvecklingen av prototypen. Först och främst skulle det avgöras om ett uppifrån-och-ner-perspektiv var att föredra framför ett från-vänster-till-höger-perspektiv och för det andra skulle det avgöras om knappar med text eller ikoner var bäst för användarvänligheten. Uppifrån-och-ner- samt från-vänster-till-höger-perspektiven valdes som grund eftersom de kändes som naturliga arbetsflöden. Dessutom skulle det testas om det var lättast att skifta vy med hjälp av flikar eller knappar. Ett annat tidigt beslut var att de tre huvudfunktionerna (skapa modeller, generera och exekvera testfall) i programmet skulle vara skilda från varandra, dvs. de tre funktionerna skulle inte visas på samma gång. Anledningen till detta beslut var att användaren skulle koncentrera sig på den funktion hon för tillfället arbetade med och slippa information från de två övriga funktionerna. Till att börja med skissades de båda prototyperna på papper för att snabbt upptäcka mindre bra designval innan onödig tid lagts ner på att implementera det i Java.

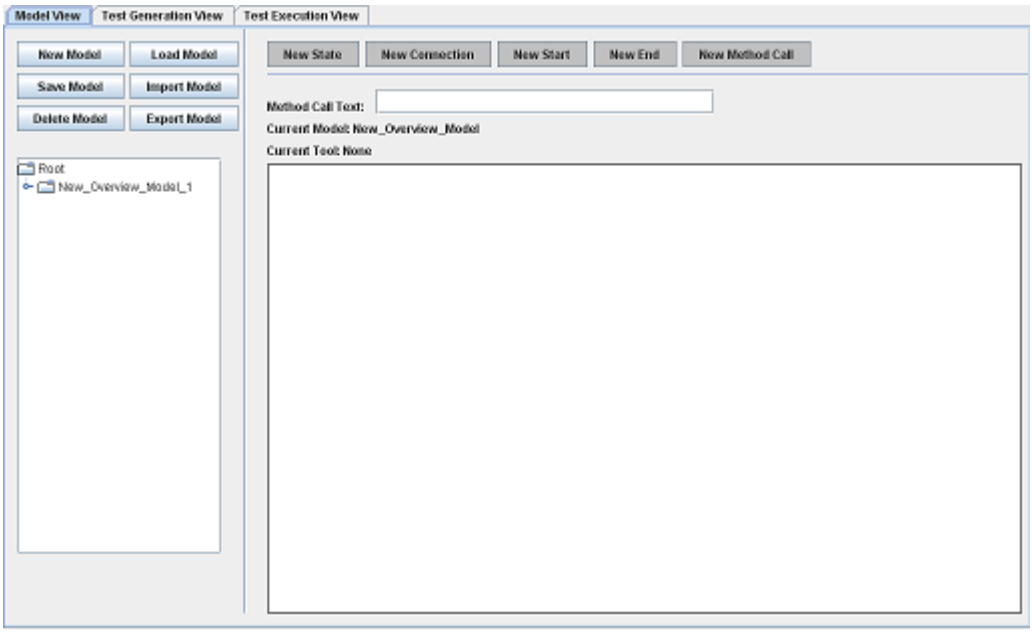
Den första prototypen, med projektnamnet Alpha, blev en prototyp som använde sig av ett uppifrån-och-ner-perspektiv och hade knappar med text istället för ikoner. Den andra prototypen, med projektnamnet Beta, blev således en prototyp som använde sig av ett

vänster-till-höger-perspektiv och knappar med ikoner istället för text. För att bläddra mellan de olika vyerna valdes ett fliksystem till prototyp Alpha och ett knappsystem i prototyp Beta.

De klasser som använts för att skapa prototyperna är i stort sett standardklasserna för att skapa användargränssnitt i Java. En del klasser behövde dock skrivas själv eftersom det inte fanns färdiga klasser till alla de designprinciper och idéer som skulle användas.

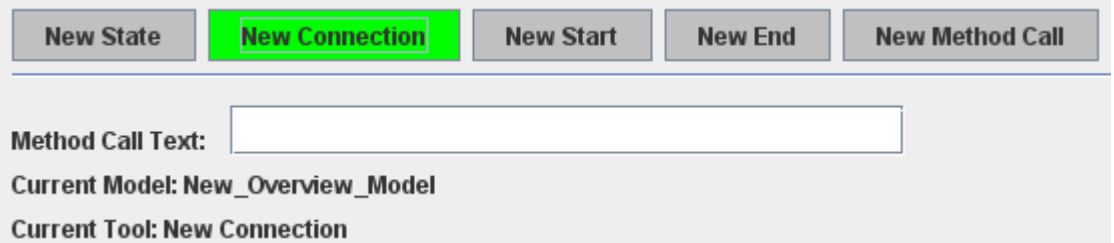
2.4.1 Prototyp Alpha

I prototyp Alpha var grundidén att användaren skulle börja arbeta från toppen av gränssnittet och sedan gå neråt i takt med att arbetet fortlöpte. Denna princip applicerades på alla de tre grundfunktionerna. Utöver detta använde Alpha sig av avgränsande streck för att sortera liknande funktionalitet på samma ställe. Knapparna i Alpha gavs en beskrivande text istället för en ikon för att undersöka om knappar med text var att föredra framför knappar med ikoner. De tre grundfunktionerna lades i varsin egen flik i programmet och det gick att växla mellan dessa via ett fliksystem där varje flik hade ett beskrivande namn.



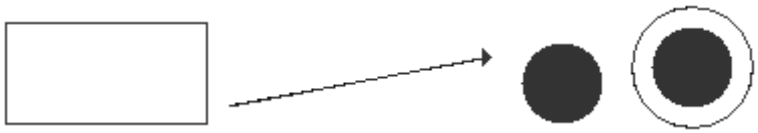
Figur 2.4.1.1: Vyn för att skapa modeller i prototyp Alpha.

I figur 2.4.1.1 ovan kan vyn för att hantera modeller i prototyp Alpha beskådas. Längst till vänster finns knappar för att bland annat spara, ladda och ta bort modeller. Under dessa finns en trädstruktursvy där det går att se hur de olika tillståndsdiagrammen är sammanlänkade. Genom att använda sig utav knapparna längst upp får användaren återkoppling i vyn undertill genom att nya tillståndsdiagram skapas och tas bort. Till höger i vyn fanns det knappar för att välja verktyg för att rita tillståndsdiagram och under dessa fanns ett fält för att ange textsträng som skulle ritas ut. Utöver detta fanns det text som berättade vilken modell och vilket verktyg som var valt i modellytan. Efter att ett verktyg valts blev verktygets knapp grönmarkerat för att tydligt synliggöra för användaren att just det verktyget blivit valt, se figur 2.4.1.2.



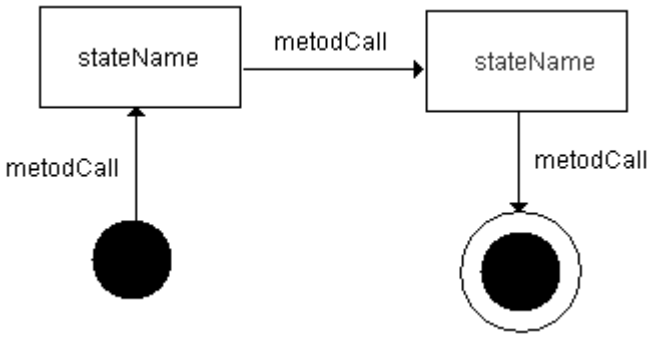
Figur 2.4.1.2: Verktygsknapp som markeras som aktiv i prototyp Alpha.

Det krävs inte mycket av en persons uppmärksamhet för att ta reda på vilket verktyg som är aktivt om det är färgkodat. Likaså ändrades färgen tillbaka till originalfärgen om verktyget inte längre var valt. Tyvärr krävdes det en nödlösning för att hantera texter i prototypen. Det fanns ingen smidig lösning för att editera texter i modellytan utan istället fick användaren placera ut textsträngarna i efterhand. Tanken var egentligen att en textsträng skulle skapas samtidigt som ett tillstånd eller en tillståndsförändring och att användaren sen, genom att klicka på texten, kunde editera den. Lösningen blev att användaren fick skriva texten som önskades i fältet ”Method Call Text” samt välja textverktyget och sedan placera ut texten i modellytan. I modellytan fungerade det på ett sådant sätt att användaren genom ett vänsterklick kunde placera ut det objekt som för tillfället var valt. Det var också möjligt att hålla inne vänsterknappen för att flytta runt objektet innan det kommit till önskad position då användaren fick släppa musknappen för att objektet skulle placeras. Objektets övre vänstra hörn började alltid vid muspekarens markör och bredde ut sig åt höger och neråt från denna position.



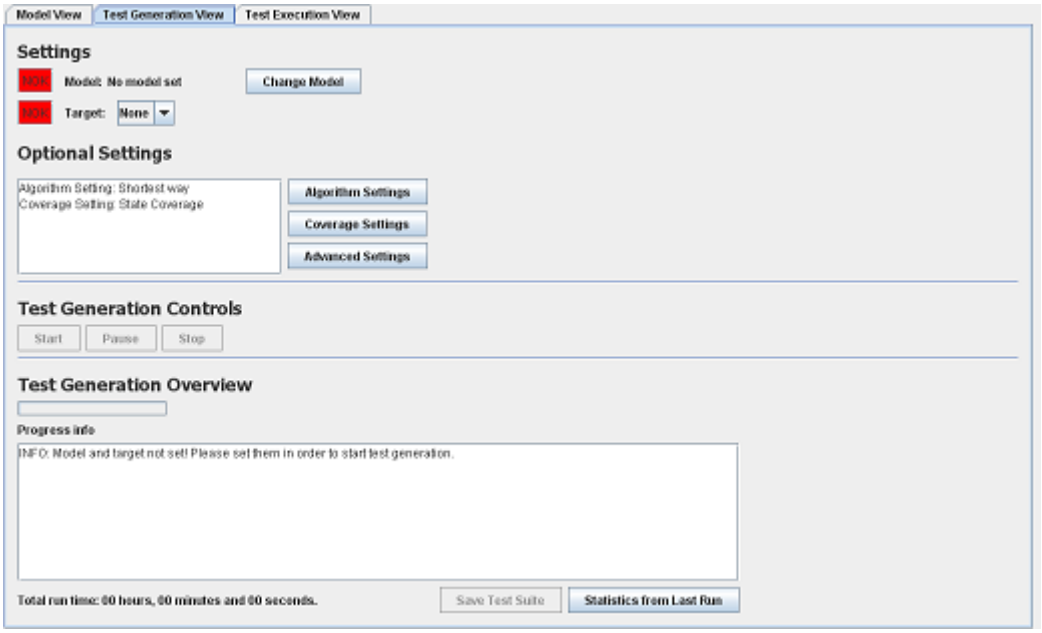
Figur 2.4.1.3: Objekten som ritades ut i modellytan. Från vänster till höger är det ett tillstånd, en tillståndsändring, ett starttillstånd samt ett sluttillstånd.

När en modell sparades så sattes den som vald modell i testgenereringsvyn. Detta val gjordes eftersom användaren med största sannolikhet ville generera testfall utifrån den modell som hon senast skapade och sparade.



Figur 2.4.1.4: Exempel på en modell som ritats i prototyp Alpha.

I vyn för att generera testfall, figur 2.4.1.5 nedan, var grundtanken att användaren först skulle göra alla obligatoriska inställningar, dvs. att välja modell och plattform.

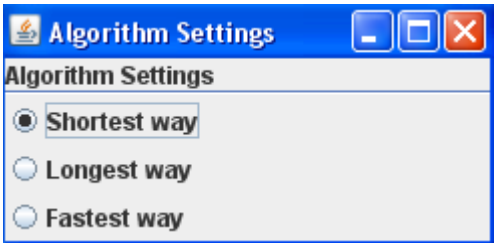


Figur 2.4.1.5: Vyn för att generera testfall i prototyp Alpha.

Efter det skulle det även vara möjligt att göra lite extra inställningar, sätta testalgoritm och modelltäckning. För att enkelt kunna identifiera om en modell eller en plattform hade valts eller inte lades två statusrutor till, en länkad till modellvalet och en till plattformsvalet. Om någon av dessa inte var valda blev statusrutan för respektive inställning röd med texten ”NOK”, som står för ”Not OK”, och om de valdes ändrades rutans färg till grön samt fick texten ”OK” för att indikera att valet genomförts. Att använda sig av färgkodningen, grön eller röd, skulle hjälpa användaren till att avgöra vilka inställningar som redan gjorts samt vilka som behövde göras.

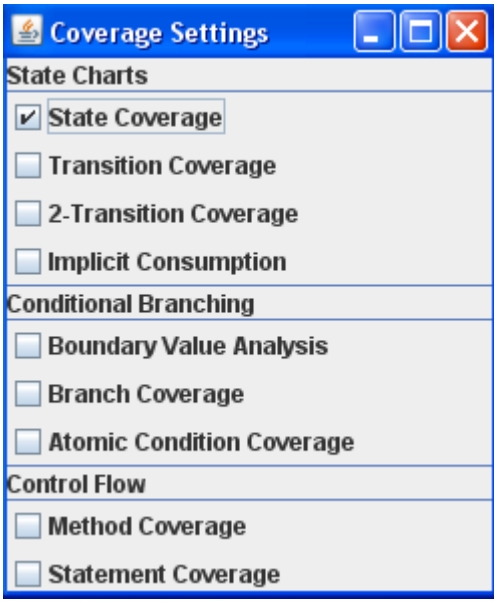
Figur 2.4.1.6: Statusrutorna i prototyp Alpha.

Samtidigt som inställningarna görs skrivs det kontinuerligt ut informativ text i textfältet under ”Test Generation Overview”. De icke obligatoriska inställningarna valdes att placeras i enskilda fönster för att endast ge användaren tillgång till de viktigaste inställningarna via huvudfönstret. För att komma åt de mindre viktiga inställningarna fick användaren klicka på knappen till inställningen och fick då upp ett nytt fönster som innehöll valmöjligheterna. Beroende på valen som genomfördes där uppdaterades informationsfönstret där algoritm- och täckningsval visades. Eftersom det bara gick att välja en algoritm i taget användes väljarknappar för att informera användaren om att endast en av algoritmerna kunde väljas åt gången.



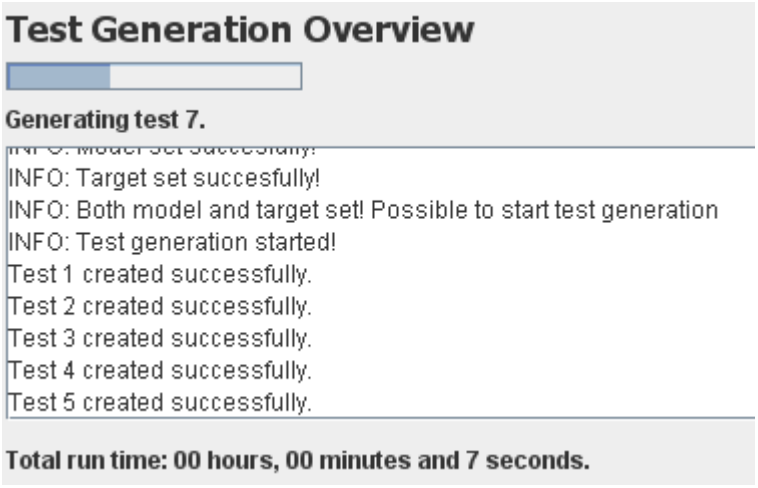
Figur 2.4.1.7: Fönstret för att välja algoritm i prototyp Alpha.

När det gällde att sätta testfallens täckning av modellen däremot så gick det att välja mer än ett alternativ. Därför användes kryssrutor för de olika alternativen och vart och ett av dessa som kryssades i täcktes således i modellen.



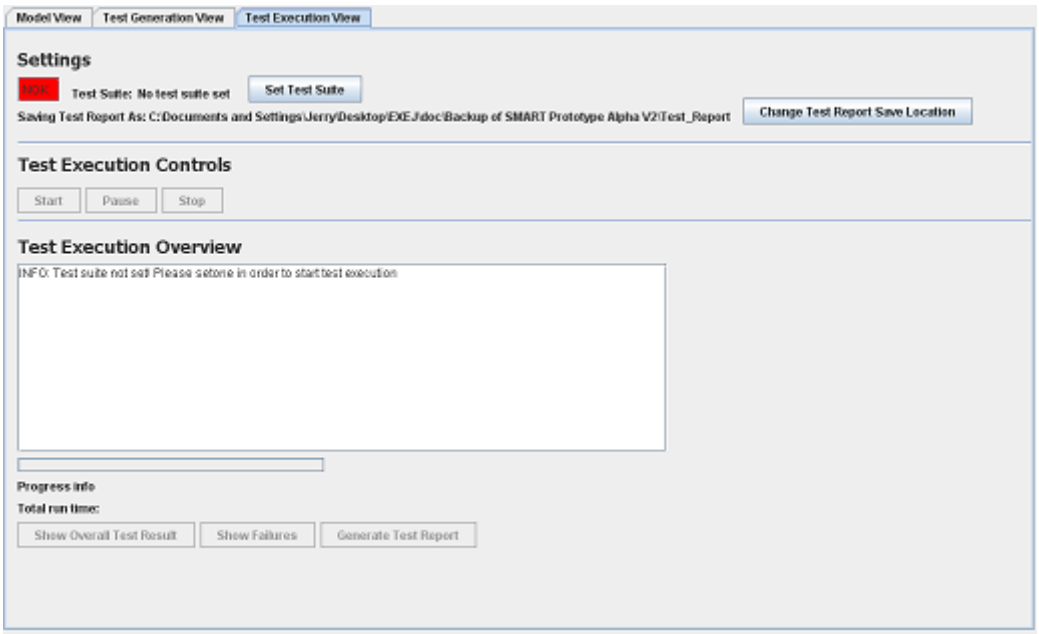
Figur 2.4.1.8: Fönstret för att välja täckning av modell i prototyp Alpha.

Efter alla inställningarna hittas knapparna för att starta, pausa och stoppa testgenereringen och detta avskiljdes tydligt från inställningarna med hjälp av ett avgränsande streck samt en fetstilt text som sa ”Test Generation Controls”. Det är inte möjligt att starta testgenereringen förrän alla obligatoriska inställningar genomförts eftersom knappen för att genomföra detta är inaktiv före det. Samma sak gäller för paus- och stoppknappen, dessa blir först aktiverade när de går att använda, dvs. efter att en testgenerering startats. Något som också bör nämnas är att det inte är möjligt att ändra modell eller plattform efter att en testgenering startats utan funktionerna för dessa val inaktiveras. Den sista delen av testgenereringsfliken innehöll en överblick av hur testgenereringen fortlöpte som gav kontinuerlig återkoppling i form av en progressbar samt text som beskrev vad som skedde.



Figur 2.4.1.9: Exempel på information som ges i prototyp Alpha när testfall genereras.

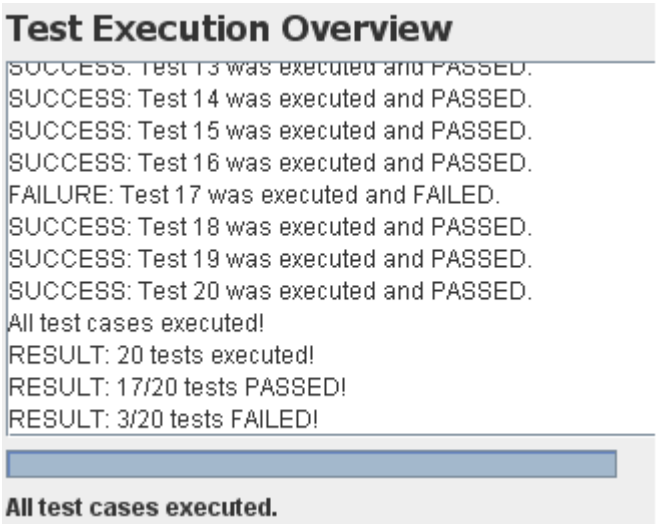
Efter att en testgenerering hade körts klart gick det att se statistik från genereringen samt att spara testsviten för senare användning. Samtidigt som en testsvit sparades sattes den som testsvit att exekvera i testexekveringsvyn eftersom användaren med största sannolikhet ville exekvera de testfall som genererats.



Figur 2.4.1.10: Vy för att exekvera testfall i prototyp Alpha.

Den sista vyn är ganska lik föregående vy med den skillnaden att istället för att generera testfall går det istället att exekvera testfall. Det första som behövs göras i denna vy är att välja en testsvit. Innan en testsvit har valts är statusrutan, som finns framför testsvitsinfon, röd och har texten ”NOK”, se figur 2.4.1.6. Det står också i textfönstret att användaren måste välja en testsvit innan det är möjligt att starta testexekveringen. Väljer användaren en testsvit ändras texten i statusrutan till ”OK” och färgen till grön för att indikera att det nu går att starta exekveringen. Samtidigt som detta skrivs det också ut information i textfönstret längst ner om att det nu är möjligt att starta körningen. Knapparna för att styra testexekveringen är

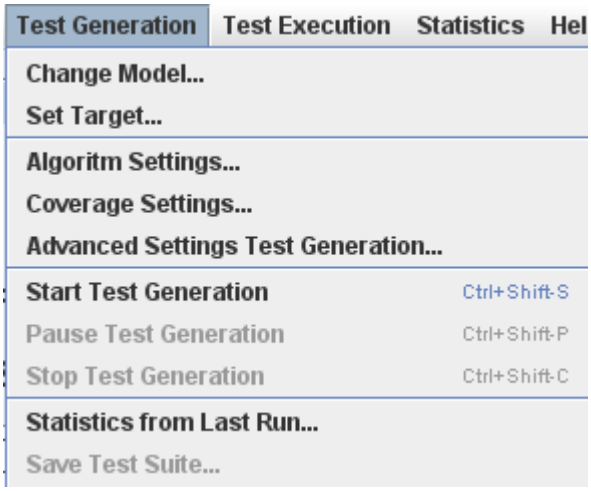
inaktiverade när de inte går att använda, t.ex. blir startknappen aktiv först efter att en testsvit valts och paus- och stoppknapparna blir aktiva först när en testexekvering startats. Längst ner i denna vy finns det en progressbar som visar hur stor del av testfallen som exekverats. Under tiden som testfallen exekveras skrivs det ut testexekveringinformation i textfältet. Beroende på om ett testfall misslyckas eller passeras skrivs olika text ut för att ge användaren återkoppling om vad som gick korrekt eller snett.



Figur 2.4.1.11: Information som skrivs ut när testfall exekveras i prototyp Alpha.

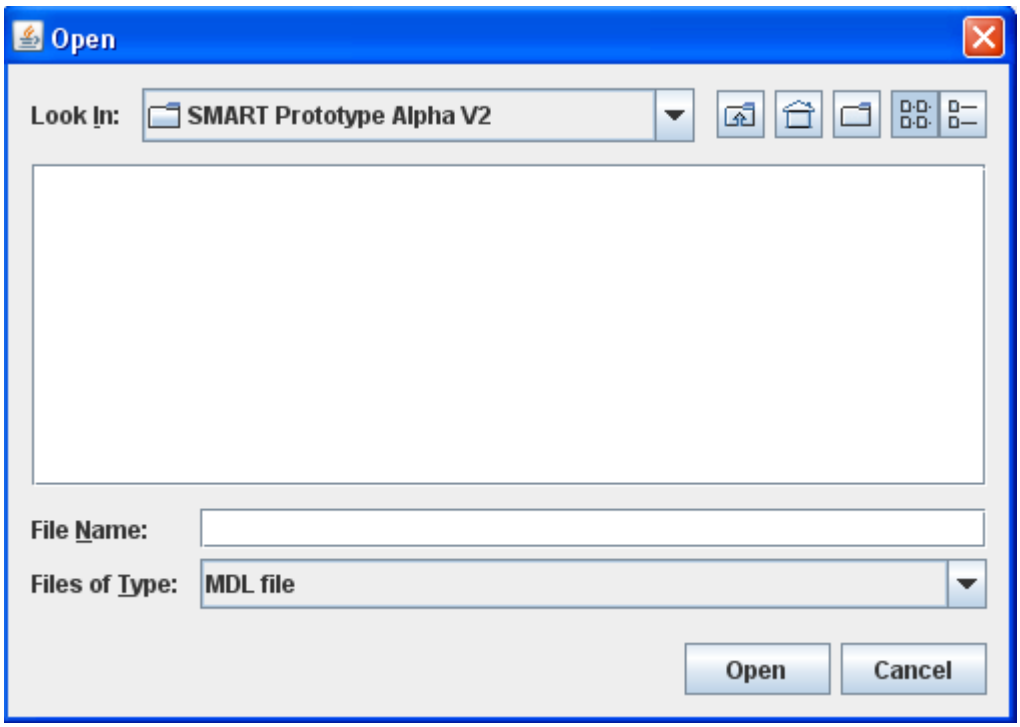
När alla testfall exekverats kan användaren välja mellan att se den övergripande testexekveringsinformation eller enbart kolla på felen. Det som är intressant ifrån en testexekvering är i stort sett bara vilka testfall som gick snett och därför skulle det ges möjlighet för användaren att fokusera sig på dessa. Det går också att spara testexekveringsinformationen till en fil för att ett enkelt dela med sig av testresultaten. Filen sparas med det namn och på det ställe som angivits under testsvitsvalet.

Förutom ovanstående vyer fanns det även ett menysystem. I detta menysystem fanns de klassiska menyalternativen ”File”, ”Edit” och ”Help” och utöver detta fanns det även ett menyalternativ för varje vy nämligen ”Model” för vyn att skapa modeller, ”Test Generation” för testgenereringsvyn och ”Test Execution” för testexekveringsvyn. I dessa tre menyalternativ fanns samma funktioner som fanns i vyerna. Detta för att ge användare som hellre använder sig av menyer möjlighet att arbeta på det sätt som de föredrar. I menyerna så var de menyalternativ som för tillfället inte gick att använda gråmarkerade, se figur nedan.



Figur 2.4.1.12: Menyn för testgenerering i prototyp Alpha.

Något som också är viktigt att påpeka var att när en fil skulle öppnas eller sparas så gjordes det i rätt katalog, hemkatalogen, och det fanns också ett filter som gjorde att rätt ändelse visades eller sparades beroende på vilken fil som önskades.



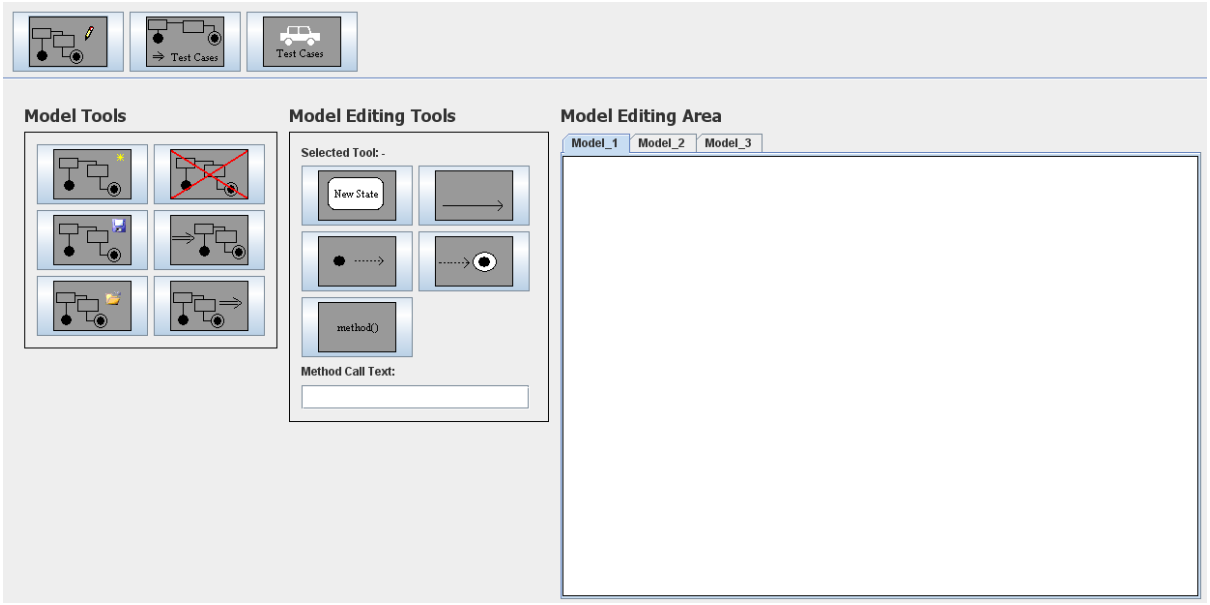
Figur 2.4.1.13: Filfilter i prototyp Alpha.

För att summera denna prototyp lite kort var liknande funktioner samlade på ett och samma område och avgränsades från övrig funktionalitet med streck. Användaren fick kontinuerligt återkoppling om vad som hände. Om det skapades en ny modell visades det i trädstrukuren genom att en fil eller mapp lades till. För själva modellerandet fick användaren återkoppling i form av att objekt ritades ut allt eftersom de placerades. Det var också tydligt för användaren vilket verktyg som var valt för tillfället då bakgrundsfärgen hos det valda verktyget var grönt medan resten av var gråa. I testgenereringsvyn skulle användaren först göra ett antal

inställningar för att sedan kunna starta testgeneringen. Under tiden som inställningarna genomfördes fick användaren kontinuerligt återkoppling genom att text och färg ändrades i inställningsytan samt genom att informativ text skrevs ut i informationsfönstret längst nere. När en testgenerering startades fick användarne också kontinuerlig återkoppling om vad som hände för att inte bli orolig om att programmet hängt sig. Det var enkelt att veta vad som var vad eftersom beskrivande text sorterade in funktionerna i olika ytor som begränsades med avgränsande streck. I testexekveringsvyn fungerade mycket som i testgenereringsvyn, återkoppling gavs i form av text och färgkodning. Genom att återanvända samma grundprincip som vid föregående vy så blev användaren inte tvingad att lära sig någon ny funktionalitet utan kunde återanvända sina kunskaper från tidigare vy.

2.4.2 Prototyp Beta

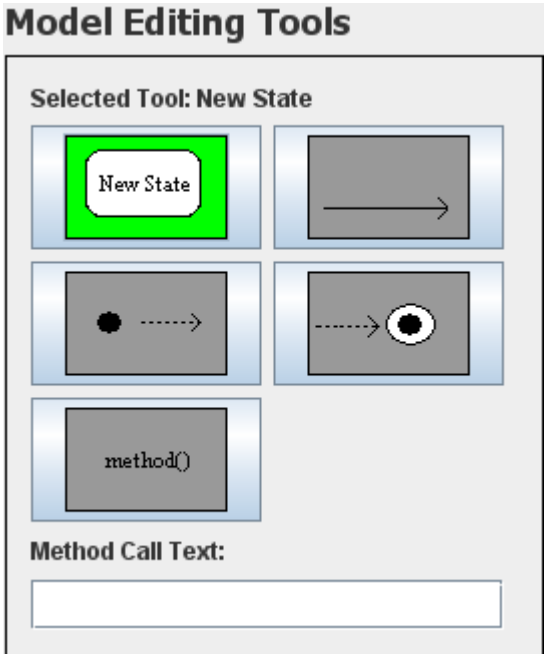
Prototyp Betas grundidé var att användaren skulle arbeta från vänster av gränssnittet och sedan fortsätta åt höger i takt med att arbetet fortlöpte. Denna princip applicerades på alla de tre grundfunktionerna. Förutom denna grundidé skulle Beta även använda sig utav ikoner för alla knappar och istället för flikar användes knappar för att växla mellan de olika vyerna. Varje vy fick således en egen knapp som gjorde det möjligt att ta sig till den specifika vyn. Att skapa modeller visades med en ikon som hade en modell och en penna i sig för att tydliggöra att det gick att rita modeller ifall den knappen användes. Den andra knappen hade en ikon med en modell och en pil som pekade mot en text som sa ”Test Cases” för att beskriva att utifrån modellen kunde testfall skapas. Den sista knappen var en bil som symboliserade att det gick att köra, exekvera, testfallen. Utöver detta använde även Beta sig utav avgränsande streck för att sortera liknande funktionalitet på samma ställe.



Figur 2.4.2.1: Vy för att skapa modeller i prototyp Beta.

Vyn för att hantera modeller såg ut som i figur 2.4.2.1 ovan. Längst till vänster fanns alla knappar som gjorde det möjligt att spara, ladda och ta bort modeller. Knapparna hade ikoner med modeller i sig där diverse detaljer ritats dit för att visa vad de olika gjorde. Exempelvis så fanns standardikonerna för att öppna och spara med i ikonerna för att spara och öppna en

modell. Det fanns tyvärr inget tydligt sätt att se relationen mellan de olika tillståndsdiagrammen men det gick att välja mellan dem i modellytan längst till höger tack vare ett fliksystem. Åtminstone fick användaren någon form av återkopplingen att något skapades eller togs bort i modellytan tack vare att flikar lades till eller togs bort. Till höger om modellknapparna fanns det knappar för att hantera tillståndsdiagramsskapandet. Precis som i Alpha blev knapparnas bakgrundsfärg gröna när ett verktyg valdes och återgick likaså till ursprungsfärgen, grå, när det inte längre var valt.

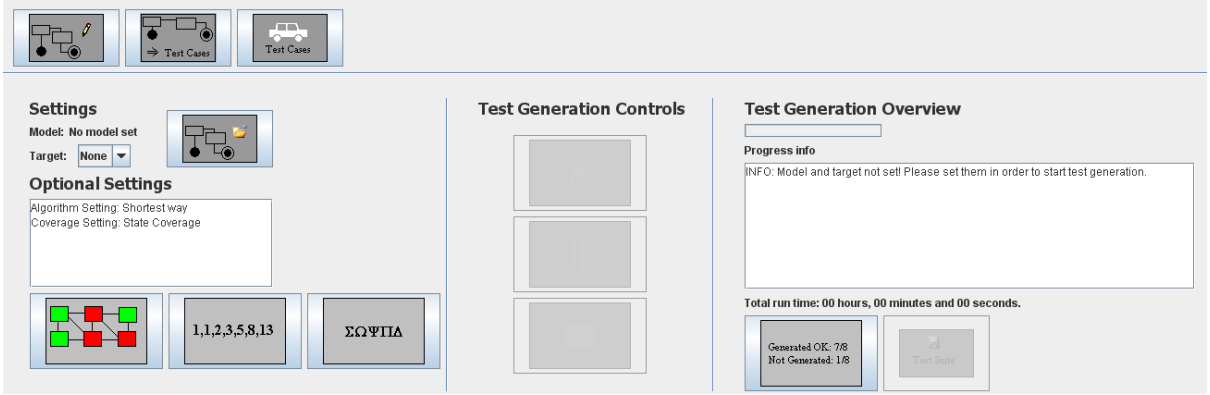


Figur 2.4.2.2: Verktyget för att rita tillstånd valt och markerat i prototyp Beta.

Det fanns också en text som beskrev vilket verktyg som var valt för tillfället. Verktygsknapparna hade ikoner som visade vad det var som skulle skapas i modellytan vilket snabbt gav användaren information om vad knapparna gjorde. För att rita modeller användes samma grundprincip som Alpha, dvs. användaren klickade i modellytan för att placera ut ett objekt. Det var också möjligt att hålla inne musknappen och flytta runt ett objekt innan det placerades. Grundidén i denna vy var att användaren först skulle skapa eller ta bort en eller flera modeller beroende på hur många hon önskade och att sedan rita i dessa med hjälp av verktygsknapparna som fanns i mitten. Längst till höger fanns sen sista anhalten, dvs. ytan för att rita modeller. Alla funktioner var sorterad inom en egen ram och hade en beskrivande text ovanför sig för att tydligt visa vilka funktioner som hörde samman.

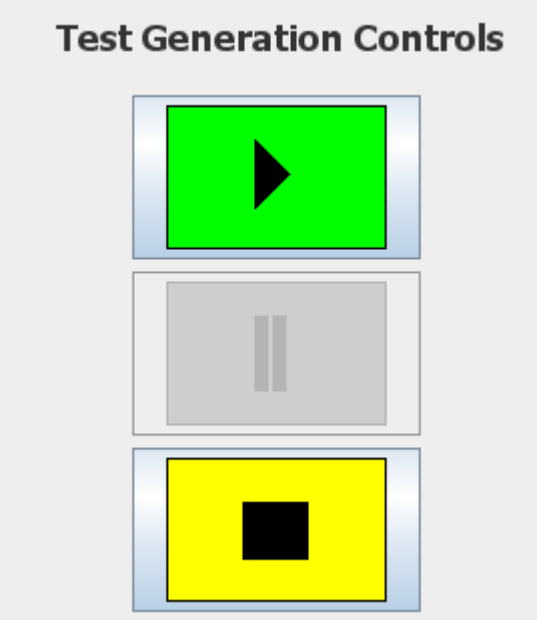
Figur 2.4.2.3: Tooltip för ett modelleringsverktyg i prototyp Beta.

Skulle det inte ges nog med information från ikonerna fanns det även tooltip som beskrev vad knapparna gjorde.



Figur 2.4.2.4: Vy för att generera testfall i prototyp Beta.

I figur 2.4.2.4 syns testgenereringsvyn. Längst till vänster fanns all funktionalitet för att göra inställningar samlade. En knapp för att öppna en modell att generera testfall från och en rullista som innehöll alla plattformar som det gick att generera testfall till. Under detta fanns en ruta där användarens val av testalgoritm och modelltäckning fanns och under denna ruta fanns knappar för att ändra denna information. Knappen för att ändra modelltäckning hade en ikon som visade en modell med gröna och röda tillstånd. Symboliken i detta skulle vara att de gröna tillstånden var de som det genererades testfall för och de röda de som det inte skapades testfall för. Den andra knappen hade en ikon som hade en bild av en matematisk algoritm som visade att det skulle vara möjligt att välja testalgoritm där. Den tredje och sista inställningsknappen hade grekiska tecken för att indikera för användaren att avancerade inställningar kunde hittas här. Till höger om inställningsfunktionaliteten fanns det knappar för att styra testgenereringen. Det gick att styra testgenereringen via en start-, en paus- och en stoppknapp som alla var inaktiva tills modell och plattform valts. Dessa tre knappar hade alla hämtat sina ikoner från den verkliga världens elektroniska prylar, t.ex. var startknappen en ”Play”-knapp hämtad från t.ex. en DVD-spelare. Ikonerna för paus och stopp var likaså hämtade från samma apparat. Startknappen hade dessutom en grön bakgrundsfärg för att indikera att det inte var något farligt med att starta testgenereringen. Paus- och stoppknapparna hade en gul bakgrundsfärg vilket skulle indikera att något skulle avbrytas.



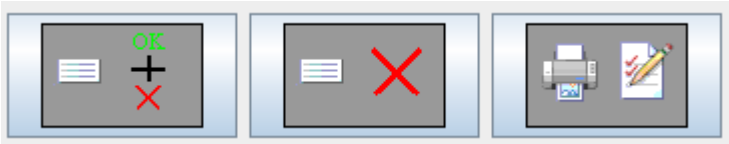
Figur 2.4.2.5: Knapparna för att styra testgenereringen i prototyp Beta.
Just denna skärmdump är tagen efter att en testgenerering pausats.

Längst till höger i denna vy hittade användaren all testgenereringsinformation. En progressbar visade hur testgenereringen fortlöpte så att användaren aldrig blev osäker om att programmet hängt sig. Likaså gavs det kontinuerligt information i textfönstret undertill om vilka testfall som genererats. Efter att alla testfall genererats färdigt var det möjligt att få statistik från testgenereringen och det gavs även möjlighet att spara testsviten för senare användning. Precis som i föregående vy fanns det tooltips på knapparna om användaren behövde mer information om vad en knapp gjorde.

Figur 2.4.2.6: Vy för att exekvera testfall i prototyp Beta.

Den tredje och sista vyn i Beta var testexekveringsvyn. Denna vy liknade testgenereringsvy ganska mycket eftersom de byggde på exakt samma koncept. Först gjordes inställningar som behövdes längst till vänster, i mitten fanns knapparna för att styra testexekveringen och längst till höger fanns informationen om hur testexekveringen fortlöpte. Den enda större skillnaden var att istället för att välja var en fil skulle sparas efter att en körning blivit klar skulle användaren istället välja var filen skulle sparas innan körningen startats och sen trycka

generera testrapport efter körningen. Ikonen för att öppna en testsvit hade tagit standardikonen för att öppna något och kombinerat den med texten ”Test Suite” för att enkelt beskriva att det var en testsvit som öppnades om användaren valde att klicka på den knappen. Ikonen för att välja var en rapport skulle sparas hade tagit standardikonen för att spara något och kombinerade denna med något som skulle likna en rapport. För att visa fullständig testexekveringsinformation användes en knapp som hade en ikon med något som liknade exekveringsinformationen kombinerat med en grön text som sade ”OK”, ett plustecken och ett rött kryss. Den gröna texten och det röda krysset skulle ge användaren information om att både testfall som gick korrekt och snett skulle visas samtidigt. Knappen för att visa enbart fel hade samma idé som föregående ikon med skillnaden att den gröna texten ”OK” och plustecknet hade tagits bort. Detta skulle indikera att endast felen visades om denna knapp användes. Längst till höger fanns en knapp för att generera en testrapport och det symboliserades med en ikon där det fanns en bild på en skrivare samt en bild på en rapport.



Figur 2.4.2.7: Knapparna som aktiveras efter att en testexekvering slutförts i prototyp Beta.

Precis som i Alpha fanns det även i Beta ett menysystem. Detta menysystem var identiskt med det som fanns i Alpha. En annan sak som var samma i Beta var filtersystemet för att spara och öppna filer med rätt ändelse.

Om denna prototyp ska summeras lite snabbt fick användaren återkoppling i form av att olika modellytor skapades eller togs bort beroende på valen längst till vänster, dvs. knapparna för att skapa nya modeller, ta bort befintliga modeller med mera. För varje vy som adderades skapades en ny flik där modellen återfanns. Likaså fick användaren återkoppling om vilket modelleringsverktyg som var valt för tillfället då dess bakgrundsfärg ändrades till grön. De tre funktionerna separerades genom tre olika rektanglar som tydligt samlade liknande funktionalitet. I testgenereringsvyn gavs det återkoppling genom att texten för modell, plattform, täckning och algoritm ändrades efter de val som gjordes och en övergripande återkoppling gavs längst till höger tack vare ett informationsfönster samt en progressbar. De olika huvudområdenas funktionalitet var samlade tillsammans och begränsades med ett streck som skilde var och en av dem från varandra. I den tredje och sista vyn, testexekveringsvyn, fungerade det mesta likadant som i testgenereringsvyn då användaren skulle återanvända arbetssättet som användes i föregående vy. Ikonerna för knapparna försökte i största möjliga mån att baseras på en konceptuell modell för att säkerställa att användaren förstod vad knappen var till för. Om det fanns ikoner som var bekanta från andra program, t.ex. en diskett för att spara, återanvändes dessa ikoner eftersom att användaren hade kunskap sedan tidigare om vad ikonen innebar.

2.5 Specificering av användbarhetsmål

Innan prototyperna skulle testas sattes ett antal användbarhetsmål upp. Målen togs fram på egen hand av författaren till denna rapport med hänsyn till de användbarhetsmått som beskrevs i litteraturavsnittet. Anledningen till att System Verification inte deltog i formuleringen av användbarhetsmålen var för att inte påverka examensarbetarens designer och funderingar. Tiden för att utföra en specifik uppgift uppskattades och andra allmänna mål för användbarheten specificerades, t.ex. andelen nya och erfarna användare som upplevde att användargränssnittet var lätt att använda. Mål för antalet fel och musklickningar per uppgift sattes också upp. Detta gjordes för att kontrollera att prototyperna inte genererade för många användarfel samt för att se om användarna behövde klicka orimligt många gånger för att slutföra en uppgift. Dessa mål låg sedan som grund för att kunna avgöra när prototypen nått ett stadium som uppfattades som lättanvänt av användarna. Likaså användes användbarhetsmålen som kontroll under utvärderingarna av prototyperna för att lättare kunna avgöra vad i gränssnittet som behövde förbättras. Visade det sig t.ex. att en uppgift tog längre tid än önskat var det uppenbart att den delen av användargränssnittet inte fungerade tillräckligt bra.

2.5.1 Allmänna användbarhetsmål

A1.1. 80 % av alla nya användare skall uppleva att programmet är lätt att använda.

A1.2. 100 % av alla erfarna användare skall uppleva att programmet är lätt att använda.

A1.3. Användarna skall kontinuerligt få återkoppling om vad som händer i programmet.

A1.4. Med hjälp av färgkodning, självinstruerande design, skall användarna kunna avgöra om något är lämpligt eller olämpligt att använda vid ett specifikt tillfälle.

A1.5. Med hjälp av avgränsande streck skall användaren kunna få en uppfattning om vilka ytor som innehåller liknande funktionalitet.

A1.6. Designen skall vara gjord på ett sådant sätt att felfrekvensen vid användandet av programmet skall ligga så nära noll som möjligt. Självfallet är det en omöjlighet att uppnå noll fel men det är åtminstone målet.

A1.7. Design skall återanvändas om det är möjligt för att användaren ska slippa att lära sig nya mönster att arbeta efter.

A1.8. Om något går fel och programmet kraschar skall det ges information på ett sådant sett att användaren kan undvika det vid nästa körning.

2.5.2 Användbarhetsmål för testmodellering

A2.1. Det skall för användaren vara möjligt att avgöra var ett objekt kommer att placeras i ritytan.

A2.2. Det skall vara uppenbart vilket verktyg som för tillfället är valt.

A2.3. Det skall inte krävas mer än maximalt två vänsterklick för att placera ut ett objekt.

A2.4. Det skall, för en ny användare, inte ta mer än 5 minuter att rita en modell som visar två tillstånd, en start, ett slut och med tillståndsbyten mellan dessa fyra objekt.

A2.5. Det skall, för en erfaren användare, inte ta mer än 2 minuter att rita en modell som

visar två tillstånd, en start, ett slut och med tillståndsbyten mellan dessa fyra objekt.

2.5.3 Användbarhetsmål för testgenerering

A3.1. Det skall, för en ny användare, inte krävas mer än 18 vänsterklick för att välja modell och plattform samt att generera testfall. Detta förutsätter givetvis att inga ytterligare inställningar måste ändras.

A3.2. Det skall, för en erfaren användare, inte krävas mer än nio vänsterklick för att välja modell och plattform samt att generera testfall. Detta förutsätter givetvis att inga ytterligare inställningar måste ändras.

A3.3. Det skall, för en ny användare, inte ta längre tid än tre minuter att välja modell och plattform för att kunna generera testfall. Tiden för att generera testfallen ingår ej i dessa tre minuter. Vidare förutsätter detta givetvis att inga ytterligare inställningar måste ändras.

A3.4. Det skall, för en erfaren användare, inte ta längre tid än en minut att välja modell och plattform för att kunna generera testfall. Tiden för att generera testfallen ingår ej i denna minut. Vidare förutsätter detta givetvis att inga ytterligare inställningar måste ändras.

2.5.4 Användbarhetsmål för testexekvering

A4.1. Det skall, för en ny användare, inte krävas mer än tolv vänsterklick för att välja testsvit samt att exekvera testfallen som finns i testsviten. Detta förutsätter givetvis att inga ytterligare inställningar måste ändras.

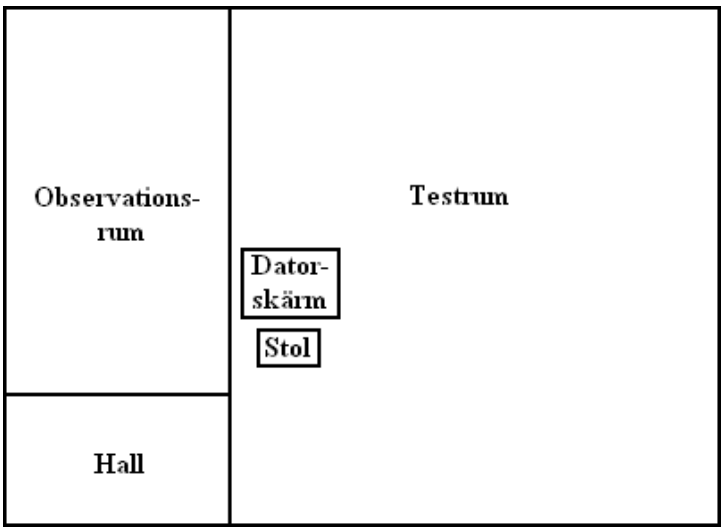
A4.2. Det skall, för en erfaren användare, inte krävas mer än sex vänsterklick för att välja testsvit samt att exekvera testfallen som finns i testsviten. Detta förutsätter givetvis att inga ytterligare inställningar måste ändras.

A4.3. Det skall, för en ny användare, inte ta längre tid än tre minuter att välja testsvit för att kunna exekvera testfallen som finns i testsviten. Tiden för att exekvera testfallen ingår ej i dessa tre minuter. Vidare förutsätter detta givetvis att inga ytterligare inställningar måste ändras.

A4.4. Det skall, för en erfaren användare, inte ta längre tid än en minut att välja testsvit för att kunna exekvera testfallen som finns i testsviten. Tiden för att exekvera testfallen ingår ej i denna minut. Vidare förutsätter detta givetvis att inga ytterligare inställningar måste ändras.

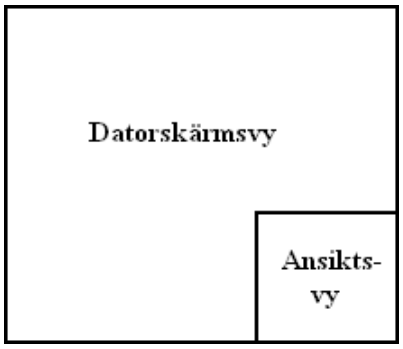
2.6 Första användartestningen av prototyperna

De två prototyperna testades av fem försökspersoner, tillhandahållna av System Verification. Inklusionskriterierna för att få ingå i den sökta målgruppen var att ha genomfört en teknisk utbildning samt att ha erfarenhet av testning eller att åtminstone ha någorlunda kännedom om teknik och testning. Försökspersonernas erfarenhet av testning behövde nödvändigtvis inte vara av modellbaserad testning. Testerna genomfördes i en användbarhetslaborationslokal på IKDC, Ingvar Kamprads Designcentrum, där utrustning för att dokumentera ljud och bild fanns att tillgå. En översikt av laborationslokalen kan ses i figur 2.6.1 nedan.



Figur 2.6.1: Användbarhetslaborationslokalen på IKDC.

Mellan observationsrummet och testrummet fanns en glasvägg som endast personer som befann sig i observationsrummet kunde se igenom. Testledaren började med att visa försökspersonen observationsrummet för att dämpa deras oro över vad som hände på andra sidan glaset. Efter det gick testledaren och försökspersonen in i testrummet där en muntlig introduktion gavs. Det som förmedlades under denna introduktion presenteras i bilaga A. En uppgiftsblankett med instruktioner för testet lästes sedan igenom av försökspersonen. Eventuella frågor om själva testet eller uppgifterna kunde ställas efter att blanketten hade lästs igenom. Uppgiftsblankettens finns att beskåda i bilaga B. När försökspersonen kände sig redo gick testledaren in i observationsrummet och knackade på fönsterrutan när inspelningen hade satts igång. Knackningen var en signal till försökspersonen att testningen av prototyperna kunde påbörjas. Det som filmades var först och främst datorskärmens yta för att se vilka handlingsval som gjordes samt testpersonens ansikte för att se eventuella reaktioner. Dessa två vyer kombinerades i en enda bild genom ett så kallat ”bild i bild”-perspektiv som var konfigurerat enligt figur 2.6.2.



Figur 2.6.2: ”Bild i bild”-perspektiv för inspelning.

Under tiden som försökspersonen testade prototyperna antecknade testledaren värdefulla kommentarer från försökspersonen samt eventuella frågor som uppstod på grund av försökspersonens agerande. Efter att de två prototyperna testats klart stoppades inspelningen och testledaren gick in i testrummet. Försökspersonen fick därpå fylla i en svarsenkät som såg ut enligt bilaga C. Det gavs även tid till att uttrycka åsikter om prototyperna angående sådant som inte togs upp i svarsenkäten. Vidare ställde testledaren frågorna som hade antecknats under testets gång för att få klarhet i varför försökspersonen agerade på ett visst sätt.

För att avgöra vilken av prototyperna som försökspersonen skulle börja med kastades en tärning. Om en etta, tvåa eller trea visades på tärningen fick försökspersonen starta med prototyp Alpha och om det blev en fyra, femma eller sexa så fick han istället börja med prototyp Beta. Det fanns dock en regel angående detta och det var att alla fem försökspersonerna inte fick starta med samma prototyp. En så jämn fördelning som möjligt önskades och det närmsta var att två personer startade med en av prototyperna och tre personer med den andra. Om det alltså, genom tärningskast, bestämdes att försöksperson ett, två och tre alla fick börja med Alpha så behövdes det inte kastas någon tärning för att avgöra vilken prototyp som försöksperson fyra och fem skulle börja med utan det blev per automatik prototyp Beta.

Ett antal attribut mättes i efterhand när det inspelade materialet studerades. Dessa attribut härstammar från Schneidermans definition av användbarhet som delar upp användbarhet i mätbara enheter. Mätdata som registrerades sammanställdes sen i en tabell för att ge en bra överblick av testresultaten. Attributen som mättes vid analysen var:

- Tid att utföra en uppgift.
- Frekvens användarfel.

Utöver detta kontrollerades det att ingen av uppgifterna krävde onödigt många musklick för att genomföras. Det exakta antalet musklickningar som användes vid varje uppgift mättes inte upp utan istället beskrivs det i resultatet om användaren klickade onödigt mycket vid en specifik uppgift. Resultaten ifrån ovanstående mätningar, svarsenkäterna samt antecknade kommentarer och frågor låg sedan till grund för att avgöra vilken av prototyperna som skulle vidareutvecklas samt vilka förbättringar som behövde göras.

Alla försökspersonerna hade ett eller flera års erfarenhet av testning och endast en person hade tidigare erfarenhet av modellbaserad testning. Samtliga försökspersoner hade någon form av teknisk utbildning bakom sig, vilket kan ses i tabell 2.6.1 nedan. För att lättare kunna

förstå varför en del klarade uppgifterna bättre i prototyp Alpha eller Beta står det också om de började med Alpha eller Beta. Anmärkning: FP är förkortning för försöksperson.

Tabell 2.6.1: Beskrivning av försökspersonerna.

Försöks-person:	**Utbildning:**	**Antal års erfarenhet av testning:**	**Tidigare erfarenhet av modellbaserad testning:**	**Började med Alpha/Beta:**
FP1	Interaktions-vetenskap	1-3 år	Nej	Alpha
FP2	Teknisk-Fysik	4-7 år	Ja	Alpha
FP3	Elektroteknik	1-3 år	Nej	Beta
FP4	Interaktions-vetenskap	1-3 år	Nej	Beta
FP5	Datavetenskap	1-3 år	Nej	Alpha

En sammanställning av resultaten ifrån svarsenkäterna finns i tabell 2.6.2 nedan:

Tabell 2.6.2: Sammanställning av svarsfrekvens ifrån svarsenkät.

Fråga 1: Antal års erfarenhet av testning?	Svarsfrekvens:
Mindre än 1 år	0.00%
1-3 år	80.00%
4-7 år	20.00%
8-15 år	0.00%
Mer än 15 år	0.00%
Fråga 2: Har du tidigare erfarenhet av modellbaserad testning?	Svarsfrekvens:
Ja	20.00%
Nej	80.00%
Fråga 3: I vilken av prototyperna var det lättast att rita en modell?	Svarsfrekvens:
Alpha	20.00%
Beta	80.00%
Fråga 4: I vilken av prototyperna var det lättast att generera testfall?	Svarsfrekvens:
Alpha	60.00%
Beta	20.00%
Fråga 5: I vilken av prototyperna var det lättast att exekvera testfallen?	Svarsfrekvens
Alpha	80.00%
Beta	0.00%

Fråga 6: Är knappar med text, användes i prototyp Alpha, eller knappar med ikoner, användes i prototyp Beta, bäst i ett modellbaserat testverktyg?	Svarsfrekvens:
Knappar med text, Alpha	0.00%
Knappar med ikoner, Beta	20.00%
En kombination av Alpha och Beta	80.00%
Fråga 7: Är ett uppifrån-och-ner-perspektiv, användes i prototyp Alpha, eller ett vänster-till-höger-perspektiv, användes i prototyp Beta, bäst för ett modellbaserat verktyg?	Svarsfrekvens:
Uppifrån-och-ner-perspektiv, Alpha	80.00%
Vänster-till-höger-perspektiv, Beta	20.00%
Fråga 8: Upplevde du att du fick nog med återkoppling från prototyperna för att förstå vad det var som hände under användandet?	Svarsfrekvens:
Ja	100.00%
Nej	0.00%
Fråga 9: Upplevde du att prototyp Alpha var lätt att använda?	Svarsfrekvens:
Ja	80.00%
Nej	20.00%
Fråga 10: Upplevde du att prototyp Beta var lätt att använda?	Svarsfrekvens:
Ja	60.00%
Nej	20.00%
Fråga 11: Om du fick välja en av prototyperna, vilken skulle du då valt baserat på den totala användarupplevelsen?	Svarsfrekvens:
Alpha	40.00%
Beta	60.00%

En uppmärksam läsare upptäcker att fråga 4 och 5 saknar 100 % svarsfrekvens. Detta beror på att en av försökspersonerna tyckte att prototyp Alpha och Beta var lika lätta att generera och exekvera testfall i. På grund av detta har denna persons svar inte tagits med i svarssammanställningen. Detsamma gäller för fråga 10 där en testperson tyckte att prototyp Beta var lätt att använda men samtidigt inte. Därför har detta svar också strukits från resultatsammanställningen.

En sammanställning av tiden samt felfrekvensen för att utföra en specifik uppgift finns i nedanstående tabeller, först presenterat som data för varje person och sedan som genomsnittlig data för alla försökspersonerna. Under en andra studie av det inspelade materialet registrerades även värdefulla kommentarer och dessa sammanfattas efter varje persons tabell. Likaså listas alla de svar som gavs till de frågor testledaren ställde efter att försökspersonerna fyllt i svarsenkäten.

Tabell 2.6.3: Data för försöksperson 1.

Person: FP1	Prototyp: Alpha	
Uppgift:	**Tid:**	**Antal fel:**
Rita och spara en modell. Motsvarar uppgift 1 och 2 på uppgiftsblanketten.	5 min och 14 s	5
Generera testfall och spara som en testfallssvit. Motsvarar uppgift 3 och 4 på uppgiftsblanketten.	1 min och 31 s	1
Exekvera testfallen i en testfallssvit. Motsvarar uppgift 5 på uppgiftsblanketten.	0 min och 38 s	0
Person: FP1	Prototyp: Beta	
Uppgift:	**Tid:**	**Antal fel:**
Rita och spara en modell. Motsvarar uppgift 1 och 2 på uppgiftsblanketten.	3 min och 45 s	0
Generera testfall och spara som en testfallssvit. Motsvarar uppgift 3 och 4 på uppgiftsblanketten.	1 min och 31 s	1
Exekvera testfallen i en testfallssvit. Motsvarar uppgift 5 på uppgiftsblanketten.	0 min och 41 s	0

Felen som FP1 gjorde var:

1. Förstod inte konceptet med att hålla inne musknappen och sen släppa upp knappen när ett objekt flyttats till önskad position.
2. Satte ut en tom text eftersom inget värde för texten hade angivits i textfältet. Resultatet blev att en tom text ritades ut. Detta fel gjordes enbart i prototyp Alpha.
3. Klickade på texten där algoritmvalet stod istället för knappen för att välja algoritm.
4. Klickade på texten där modellvalet stod istället för knappen för att välja modell.

Värdefulla kommentarer samt svar från FP1 på frågorna som ställdes av testledaren:

1. Att flera modeller var skapade vid start kändes ologiskt. Det hade räckt att endast en var skapad vid start.
2. Ikonknapparna som hade tooltip var väldigt bra när det skulle skapas modeller. Försökspersonen tyckte även att det ibland hade varit bra att kombinera ikon och text i en knapp.
3. Upplevde att ikonknappar var bättre än text för start, paus och stopp.
4. Knapparna för start och paus skulle kombinerats till en knapp.
5. Ville ha en Ok- och en Cancel-knapp i algoritm- och täckningsfönsterna.

6. Ikonerna i Beta för att spara modell och dylikt var alldeles för detaljerade. Det blev svårt att se vad knapparna gjorde. Det hade räckt med de standardiserade ikonerna, t.ex. en diskett för att spara.
7. Saknade möjligheten att flytta objekt som skapats genom att klicka på dem. Samtidigt vore det bra om det gick att ångra senaste ritade objekt samt ångra det som ångrats. Stöd för att kopiera, klippa ut och klistra in skulle också vara uppskattat.
8. Texten för vilket verktyg som var valt känns överflödigt då verktygsvalet lätt identifieras via färgkodning på knapparna.
9. Föreslog att det skapades en textruta i tillstånd och vid tillståndsändringar när de initierades. Dessa textrutor skulle det sen gå att editera.
10. Trots ovanstående förslag tyckte försökspersonen att textediteringen fungerade smidigt när konceptet satt sig.
11. Tillståndsändringar mellan objekt ska låsas till objekten så att det tydligt framgår mellan vilka objekt som tillståndsändringarna är.
12. Ett förslag att hålla inne shift-knappen och sen rita en tillståndsändring mellan två objekt gavs. Det fungerade på ett sådant sätt att när knapparna släpptes upp ritades tillståndsändringen ut med raka streck istället för så som personen ritat.
13. En överblick över vilka testfall som ingick i en testsvit saknades. Det skulle varit väldigt bra för att veta vilka testfall som exekverades. I en sådan överblick skulle det även ges möjlighet att välja eller välja bort flera eller enstaka testfall.
14. Lägga till knappkombinationer för att snabbt kunna byta verktyg vid modellritande.
15. Gillade bäst att navigera mellan vyer med flikar. Att navigera mellan olika vyer var svårare med knappar.
16. Föredrog uppifrån-och-ner-perspektivet i prototyp Alpha före vänster-till-höger-perspektivet i prototyp Beta. Kändes som ett mer inarbetat arbetssätt, som att arbeta av en lista.
17. Tyckte det gavs bra återkoppling tack vare att det skrevs ut text vid generering och exekvering av testfall.
18. Upplevde att det var mycket lättare att rita en modell i Beta eftersom konceptet var bekant efter att ha testat Alpha.
19. Tyckte att ikonerna hjälpte till när det gällde att rita en modell, försvårade bara när tester skulle genereras eller exekveras.

Anmärkning: FP1 behövde starta om ritandet av modell i prototyp Alpha en gång. Trots detta var han en av de som ritade modell på kortast tid och med minst fel. Dock glömde han att sätta ut text i sin första modell vilket med största sannolikhet skulle dragit upp tiden och antalet fel som gjordes.

Tabell 2.6.4: Data för försöksperson 2.

Person: FP2	Prototyp: Alpha	
Uppgift:	**Tid:**	**Antal fel:**
Rita och spara en modell. Motsvarar uppgift 1 och 2 på uppgiftsblanketten.	9 min och 22 s	7
Generera testfall och spara som en testfallssvit. Motsvarar uppgift 3 och 4 på uppgiftsblanketten.	2 min och 52 s	1
Exekvera testfallen i en testfallssvit. Motsvarar uppgift 5 på uppgiftsblanketten.	0 min och 57 s	0
Person: FP2	Prototyp: Beta	
Uppgift:	**Tid:**	**Antal fel:**
Rita och spara en modell. Motsvarar uppgift 1 och 2 på uppgiftsblanketten.	5min och 51 s	3
Generera testfall och spara som en testfallssvit. Motsvarar uppgift 3 och 4 på uppgiftsblanketten.	2 min och 35 s	0
Exekvera testfallen i en testfallssvit. Motsvarar uppgift 5 på uppgiftsblanketten.	1min och 41 s	0

Felen som FP2 gjorde var:

1. Försökte att rita i modellytan utan att ha valt ett verktyg.
2. Klickade endast en gång för att rita en tillståndsändring. Det krävs att användaren håller inne musknappen och flyttar muspekaren till önskad slutposition för att en tillståndsändring ska skapas.
3. Ville rita en tillståndsändring från och till samma objekt men ritade fel.
4. Klickade en gång på knappen för att sätta ut text och sen en gång till på samma knapp efter att text hade skrivits in. För att skriva ut flera texter i rad behövs inte verktyget väljas om.
5. Klickade på knappen för verktyget som skapar tillståndsändringar trots att det redan var valt.
6. Skrev in text i textfältet men glömde att välja textverktyget före han försökte att placera det i ritytan.
7. Sparade modellen som ett projekt istället för modell. Hittade aldrig alternativet för att spara modell.

Värdefulla kommentarer och svar från FP2 på frågorna som ställdes av testledaren:

1. Saknade möjligheten att flytta objekt som skapats genom att klicka på dem. Samtidigt vore det bra om det gick att ångra senaste ritade objekt samt ångra det som ångrats. Stöd för att kopiera, klippa ut och klistra in skulle också vara uppskattat. Knappen *Delete* ska gå att trycka för att radera markerat objekt.
2. Tillståndsändringen mellan två objekt ska låsas till objekten så att det tydligt framgår mellan vilka objekt som tillståndsändringen är. Tillståndsändring till samma objekt, så kallad återkoppling, ska också kunna låsas till sig själv.
3. När en algoritm valdes i algoritmfönstret borde fönstret stängas ner.
4. Önskade att det fanns en Ok- samt en Cancel-knapp i täckningsfönstret.
5. Tyckte att avancerade valmöjligheter kändes överflödiga.
6. När en modell ska raderas, fråga användaren om hon verkligen vill radera modellen så att en modell inte raderas av misstag.
7. Skapa en textruta i ett tillstånd eller vid en tillståndsändring när de skapas och som sedan går att editera istället för att sätta ut texterna i efterhand. Svårt att identifiera till vilket objekt texten var länkad till i nuvarande prototyper.
8. Ett verktyg för att markera flera objekt hade varit bra om flera objekt ska flyttas.
9. Tyckte att det var bra att den modell som sparades sattes som modell att generera testfall ifrån. Samma gällde när en testsvit sparades och den sattes som testsvit att exekveras.
10. Ogillade starkt knapparna för att växla vy. Föredrog flikarna i Alpha.
11. Upptäckte att när en testsvit skulle öppnas så stod det spara i fildialogen. Självklart ett fel från programmerarens sida.
12. Ett antal ikoner var inte lätta att förstå men tooltipen hjälpte till att förstå vad knapparna gjorde.
13. Lägg till knappar efter varje täckningsalternativ som på något sätt gör att en kort beskrivning av vad de olika täckningsalternativen innebär visas.
14. Önskvärt att kunna högerklicka i modellytan för att snabbt kunna ändra verktyg eller för att använda sig av de vanliga valen för att editera, t.ex. klippa ut, klistra in och ångra.
15. Föredrog uppifrån-och-ner-perspektivet i prototyp Alpha före vänster-till-höger-perspektivet i prototyp Beta. Kändes mest naturligt.
16. Gillade den tydliga återkopplingen med text som skrevs ut vid generering och exekvering av testfall.

Anmärkning: FP2 behövde starta om ritandet av modell i prototyp Alpha en gång.

Tabell 2.6.5: Data för försöksperson 3.

Person: FP3	Prototyp: Beta	
Uppgift:	**Tid:**	**Antal fel:**
Rita och spara en modell. Motsvarar uppgift 1 och 2 på uppgiftsblanketten.	8 min och 7 s	6
Generera testfall och spara som en testfallssvit. Motsvarar uppgift 3 och 4 på uppgiftsblanketten.	6 min och 57 s	8
Exekvera testfallen i en testfallssvit. Motsvarar uppgift 5 på uppgiftsblanketten.	0 min och 28 s	0
Person: FP3	Prototyp: Alpha	
Uppgift:	**Tid:**	**Antal fel:**
Rita och spara en modell. Motsvarar uppgift 1 och 2 på uppgiftsblanketten.	4 min och 38 s	1
Generera testfall och spara som en testfallssvit. Motsvarar uppgift 3 och 4 på uppgiftsblanketten.	1 min och 14 s	1
Exekvera testfallen i en testfallssvit. Motsvarar uppgift 5 på uppgiftsblanketten.	0 min och 35 s	0

Felen som FP3 gjorde var:

1. Klickade endast en gång för att rita en tillståndsändring. Det krävs att användaren håller inne musknappen och flyttar muspekaren till önskad slutposition för att en tillståndsändring ska skapas.
2. Satte ut en tom text eftersom inget värde för texten hade givits i textfältet så en tom text ritades ut. Glömde att sätta ett värde för texten efter att textverktyget valts.
3. Försökte flytta text som hade satts ut men satte istället ut flera texter.
4. Klickade på knappen för ett verktyg som redan var valt.
5. Sparade modellen som ett projekt istället för modell. Hittade aldrig alternativet för att spara modell.
6. Lyckades inte hitta knappen för att byta vy till testgenerering. Testledaren var tvungen att visa knappen och dess funktion.
7. Klickade på texten där algoritmvalet stod istället för knappen för att välja algoritm.
8. Klickade på texten där modellvalet stod istället för knappen för att välja modell. Lyckades inte hitta knappen för att välja modell i Beta.
9. Tryckte på startknappen i testgenereringsvyn trots att modell och plattform inte valts samt det faktum att knappen var inaktiv.
10. Tryckte på startknappen i testgenereringsvyn efter att plattform hade satts trots att det fortfarande krävdes att en modell sattes.

Värdefulla kommentarer och svar från FP3 på frågorna som ställdes av testledaren:

1. Saknade möjligheten att flytta objekt som skapats genom att klicka på dem. Samtidigt vore det bra om det gick att ångra senaste ritade objekt samt ångra det som ångrats. Stöd för att kopiera, klippa ut och klistra in skulle också vara uppskattat.
2. Tillståndsändringen mellan två objekt ska låsas till objekten så att det tydligt framgår mellan vilka objekt som tillståndsändringen är. Tillståndsändring till samma objekt, så kallad återkoppling, ska också kunna låsas till sig själv. Gärna använda ett klick för start och ett andra klick för slutet av återkopplingen.
3. Ville få återkoppling i informationsrutan när en testsvit sparas.
4. Skapa en textruta i ett tillstånd eller vid en tillståndsändring när de skapas och som sedan går att editera istället för att sätta ut texterna i efterhand. Svårt att identifiera till vilket objekt texten var länkad till i nuvarande prototyper. Upplevde också att det var svårt att identifiera exakt var texten skulle hamna.
5. Ogillade starkt knapparna för att växla vy. Föredrog flikarna i Alpha.
6. Föredrog uppifrån-och-ner-perspektivet i prototyp Alpha före vänster-till-höger-perspektivet i prototyp Beta. Hade stött på det i många tidigare program.
7. Den kontinuerliga textinformationen för testgenerering och testexekvering gav tydlig återkoppling av vad som hände.
8. Tyckte att Beta var bättre att arbeta i efter att man lärt sig ikonernas funktionalitet. Kanske tog längre tid att lära sig men i slutändan skulle den vara mest effektivt.
9. Ville ha någon form av återkoppling när en modell laddades då inget hände.
10. Tyckte det var svårt att se var ett objekt skulle hamna. Skulle istället velat ha en sorts av förhandsvisning istället för att hålla inne musknappen.
11. Önskade att det fanns någon form av förklaring för de olika täckningsalternativen innebar, t.ex. en informationsknapp som när den klickades visade info om täckningen.
12. Gillade inte ikonerna för täcknings- och algoritmfönsterna i Beta.

Tabell 2.6.6: Data för försöksperson 4.

Person: FP4	Prototyp: Beta	
Uppgift:	**Tid:**	**Antal fel:**
Rita och spara en modell. Motsvarar uppgift 1 och 2 på uppgiftsblanketten.	7 min och 14 s	5
Generera testfall och spara som en testfallssvit. Motsvarar uppgift 3 och 4 på uppgiftsblanketten.	8 min och 16 s	6
Exekvera testfallen i en testfallssvit. Motsvarar uppgift 5 på uppgiftsblanketten.	0 min och 26 s	0

Person: FP4	Prototyp: Alpha	
Uppgift:	**Tid:**	**Antal fel:**
Rita och spara en modell. Motsvarar uppgift 1 och 2 på uppgiftsblanketten.	5 min och 53 s	0
Generera testfall och spara som en testfallssvit. Motsvarar uppgift 3 och 4 på uppgiftsblanketten.	2 min och 57 s	0
Exekvera testfallen i en testfallssvit. Motsvarar uppgift 5 på uppgiftsblanketten.	0 min och 44 s	0

Felen som FP4 gjorde var:

1. Valde textverktyget men missade att skriva text i textfältet. Därmed placerades det ut en tom text med textverktyget.
2. Klickade på textverktygets knapp mer än en gång för att sätta ut flera texter fastän att verktyget är valt även efter att ha satt ut en text.
3. Missade att välja plattform och därmed att generera testfall i Beta eftersom knappen för att välja modell fångade användarens fokus. Han såg det dock i efterhand men genererade inte heller då testfallen.
4. Använde ”Load Model” för att försöka sätta modell men funktionen ”Load Model” öppnar bara en modell i modellytan.

Värdefulla kommentarer och svar från FP4 på frågorna som ställdes av testledaren:

1. Ville ha möjlighet att flytta objekt eller text som skapats.
2. Ville ha möjlighet att ta bort objekt som skapats.
3. Tyckte att det var smidigt att hålla inne musknappen och sen släppa musknappen där ett objekt skulle placeras.
4. Tyckte att det var jobbigt att behöva gå till textboxen och sen tillbaka till modellytan varje gång som en text skulle läggas till. Tyckte däremot att sättet att lägga till texter var smidigt i övrigt.
5. Tyckte att ikonerna i Beta var för detaljerade och för snarlika varandra. Det skulle räckt med standardikonerna för att spara och dylikt, dvs. ta bort modelldetaljerna från ikonerna. Dessutom tyckte FP4 att knapparna i Beta var lite väl stora.
6. Ville ha knappar för att acceptera eller ångra valen som gjordes i täckningsfönstret och algoritmfönstret.
7. Tyckte att det var smidigt att kunna se endast fel från en testexekvering.
8. Gillade inte knapparna för att byta vy i Beta, hade svårt att hitta dem till en början.
9. Ville ha möjligheten att döpa om modeller.
10. Tyckte att det var dåligt att det inte fanns några tooltip till knapparna i Alpha.
11. Tyckte att modellknapparna med text som användes i Alpha fungerade mycket sämre än modellknapparna med ikoner som användes i Beta.

12. Tyckte att det skulle visats ett felmeddelande när en pil ritats ut felaktigt. En enkel lösning hade varit att rödmarkera sådant som inte är rätt placerat, t.ex. pilar som ritats ut utan att ha kopplats till andra objekt. En annan lösning hade varit att endast göra det möjligt att skapa pilar som länkas till objekt.
13. Tyckte att det var bra med färgkodning för att visa vilket verktyg som var aktivt för tillfället. Likaså var färgkodningen bra för att visa om något var valt eller inte, rött för ej valt och grönt för valt.
14. Ett verktyg för att markera ett eller flera objekt för att sedan kunna flytta dem hade varit bra.
15. Tyckte att ”Show Overall Information” och ”Show Failures” skulle vara en gemensam knapp som skiftade text beroende på vad som visades för tillfället.
16. Ville ha knappkombinationer för att skifta mellan modellverktygen.
17. Önskade att det fanns någon form av förklaring för de olika täckningsalternativen, t.ex. en informationsknapp som när den klickades visade information om täckningen.
18. Ville att endast de viktigaste funktionerna skulle visas i menyerna.
19. Tyckte att det skulle vara bra med en översikt av modellen där det visades hur stor del av modellen som täcktes av de testfall som fanns i den testsvit som öppnats.
20. Upplevde att arbetsflödet i prototyp Alpha var mer naturligt än det i prototyp Beta.

Tabell 2.6.7: Data för försöksperson 5.

Person: FP5	Prototyp: Alpha	
Uppgift:	**Tid:**	**Antal fel:**
Rita och spara en modell. Motsvarar uppgift 1 och 2 på uppgiftsblanketten.	8 min och 19 s	22
Generera testfall och spara som en testfallssvit. Motsvarar uppgift 3 och 4 på uppgiftsblanketten.	1 min och 46 s	1
Exekvera testfallen i en testfallssvit. Motsvarar uppgift 5 på uppgiftsblanketten.	0 min och 21 s	0
Person: FP5	Prototyp: Beta	
Uppgift:	**Tid:**	**Antal fel:**
Rita och spara en modell. Motsvarar uppgift 1 och 2 på uppgiftsblanketten.	4 min och 38 s	2
Generera testfall och spara som en testfallssvit. Motsvarar uppgift 3 och 4 på uppgiftsblanketten.	1 min och 14 s	1
Exekvera testfallen i en testfallssvit. Motsvarar uppgift 5 på uppgiftsblanketten.	1 min och 4 s	1

Felen som FP5 gjorde var:

1. Försökte att rita objekt i modellytan utan att först ha valt ett verktyg.
2. Hade problem med att se sambandet mellan textfältet och textverktyget.
3. Använde ”Load Model” för att sätta modell men funktionen ”Load Model” öppnar bara en modell i modellytan.
4. Satte platsen som testrapporten skulle genereras till när han skulle sätta vilken testsvit som skulle exekveras.

Värdefulla kommentarer och svar från FP5 på frågorna som ställdes av testledaren:

1. Skulle vilja att det fanns möjlighet att flytta objekt.
2. Hade velat att det klickades en gång för att markera starten av en tillståndsändring och en andra gång för att markera slutet. Dessutom ville han att de skulle länkas till objekten så att det tydligt framgick emellan vilka objekt som kopplingen var. Att klicka två gånger hade löst problemet med att länka tillbaka till sig själv.
3. Vill att någon form av meny dök upp om det högerklickades i modellytan.
4. Tyckte att det var väldigt besvärligt att sätta ut text. Istället skulle det gå att markera ett tillstånd eller en tillståndsförändring och sen få upp en textruta där det gick att skriva texten. Ett annat alternativ skulle vara att högerklickning gjorde det möjligt att skriva text.
5. Ville ha knappar för att acceptera eller ångra valen som gjordes i täckningsfönstret och algoritmfönstret.
6. Tyckte att det var svårt att förstå statusrutorna som visade om något var satt eller inte.
7. Upplevde det som ologiskt att hålla inne vänsterknappen på musen för att flytta runt ett objekt innan det placerades.
8. Föredrog knappar med ikoner framför knappar med text.
9. Gillade inte ikonerna för täcknings- och algoritmfönsterna.
10. Tyckte att det var mycket lättare att använda Beta eftersom han lärde sig av sina misstag i Alpha. Detta gällde de element som var samma i båda prototyperna.
11. Tyckte att tooltipen i prototyp Beta var till stor hjälp när ikonerna gav dålig information om funktionen.
12. Föredrog från-vänster-till-höger-perspektivet eftersom västerlänningar läser på det sättet.
13. Gillade att knapparna markerades med färgkodning efter att de valts.
14. Uppskattade textinformationen som visades under textgenereringen och testexekveringen.

Tabell 2.6.8: Genomsnittlig data per person.

Genomsnittlig data per person	Prototyp: Alpha	
Uppgift:	**Tid:**	**Antal fel:**
Rita och spara en modell. Motsvarar uppgift 1 och 2 på uppgiftsblanketten.	6 min och 41,2 s	7
Generera testfall och spara som en testfallssvit. Motsvarar uppgift 3 och 4 på uppgiftsblanketten.	2 min och 4 s	0,8
Exekvera testfallen i en testfallssvit. Motsvarar uppgift 5 på uppgiftsblanketten.	0 min och 44 s	0
Genomsnittlig data per person	Prototyp: Beta	
Uppgift:	**Tid:**	**Antal fel:**
Rita och spara en modell. Motsvarar uppgift 1 och 2 på uppgiftsblanketten.	5 min och 55 s	3,2
Generera testfall och spara som en testfallssvit. Motsvarar uppgift 3 och 4 på uppgiftsblanketten.	4 min och 6,6 s	3,2
Exekvera testfallen i en testfallssvit. Motsvarar uppgift 5 på uppgiftsblanketten.	0 min och 52 s	0,2

2.6.1 Slutsatser från första testningen

Det går snabbt att dra lite slutsatser från tabellen ovan och det är att det var lättast att rita i prototyp Beta medans det var lättare att generera och exekvera testfall i Alpha, sistnämnda dock marginellt. Till de centrala frågeställningarna kunde följande svar ges:

1. Uppifrån-och-ner-perspektivet var bättre än vänster-till-höger-perspektivet eftersom de flesta tyckte att det var mer logiskt att arbeta på det sättet. Jämför med att arbeta av en lista med deluppgifter där uppgifterna i slutet kräver att de första blivit färdiga.

2. Om det skulle vara ikoner eller text på knappar var det lite mer skilda meningar om men det som kan framläsas ur enkätsvaren samt kommentarerna är att en kombination av de båda hade varit att föredra. Testarna ville att knapparna för ritverktygen i modellvyn samt start-, paus- och stoppknapparna skulle ha ikoner. De funktioner som det inte fanns logiska ikoner eller kopplingar till skulle ha text istället för ikoner.

3. Allihop föredrog att navigera mellan vyerna med hjälp av fliksystemet framför att byta mellan vyerna med knappar. En anledning till detta kan ha varit att ikonerna till knapparna för att byta vy inte var tillräckligt informativa för testpersonerna.

En lista över de positiva och mindre positiva detaljerna hos prototyperna finns nedan. Listan över de mindre positiva detaljerna användes senare som checklista för att veta vad som skulle ändras till den vidareutvecklade prototypen.

Positivt hos prototyperna:

- Tooltip fungerade väldigt bra för knappar med ikoner.

- Den kontinuerliga återkopplingen, som gavs via progressbar och informationslogg, när testfall genererades och exekverades var väldigt uppskattat och gav användaren information om att något skedde. Annars skulle det lätt kunna uppstå förvirring om ingen återkoppling hade getts.

- Om en modell sparas är det bra att den sätts som modell i testgenereringsvyn eftersom användaren troligtvis vill generera testfall utifrån denna modell. Samma sak gällde när en testsvit sparades.

- Färgkodningen hjälpte till att avgöra vilket ritverktyg som för tillfället var valt. Samma sak gällde statusrutorna för om modell och plattform var satt eller inte, rött indikerade att något var fel och grönt att något var korrekt.

- Bra att det fanns möjlighet att växla mellan att visa fullständig informationslogg från testexekveringen eller att endast visa felen från exekveringen.

Förslag på förbättringar inför nästa version:

- Textverktyget fungerade inte bra. Användarna trodde att det räckte med att välja verktyget och sen klicka i modellytan. Efter det skulle det dyka upp en textruta som det gick att skriva i. En lösning på detta hade varit om en ruta skapades i tillstånden och vid tillståndsändringarna som det sen gick att klicka i och editera texten via. Detta används redan i många program så det hade känts mest naturligt eftersom det är en inarbetad metod.

- Försökspersonerna tyckte inte att tillståndsändringarna gav tillräckligt med återkoppling ifall en tillståndsändring länkades mellan två objekt eller inte. I de nuvarande prototyperna kunde tillståndsändringarna placeras en till flera pixlar ifrån ett objekt eller rentav inuti ett objekt. Istället skulle det varit bra om tillståndsändringarna kände av detta. Om så var fallet skulle tillståndsändringen placerats precis vid gränsen av objekten så att det verkligen markerades mellan vilka objekt tillståndsändringen sattes.

- Konceptet för att placera objekt i modellytan fungerade inte så bra som det var tänkt. Att det gick att hålla inne musknappen och sen flytta objektet till önskad position upptäcktes aldrig av en del av försökspersoner. Istället skulle det varit bra om det gavs en form av förhandsvisning av var objektet hamnade före det placerades.

- Onödigt att visa både start- och pausknappen samtidigt då det inte går att använda dem på samma gång. Istället borde dessa knappas kombineras till en enda som ändras beroende på om testgenerering eller exekvering startats eller ej.

- Alla testarna ville att det skulle finnas stöd för att flytta objekt. Ett bra förslag var att ge användarna tillgång till ett verktyg som lät dig markera en yta och sen flytta allt som fanns innanför området. För att flytta ett objekt skulle det räcka med att klicka på objektet i fråga och sen skulle det vara möjligt att flytta det.

- Alternativ för att ångra, ångra det som ångrats, kopiera, klistra in samt ta bort borde finnas. De tre sistnämnda bör även kunna användas för en markerad yta. Ett annat sätt att komma åt dessa alternativ, förutom knappkombinationer eller via menyn, skulle varit om användaren högerklickade i modellvyn och en lista med dessa alternativ dök upp.

- I fönstret där det går att välja algoritm hade det varit bra om det fanns knappar för att acceptera samt neka valen, t.ex. ”OK” och ”Cancel”. Dessa knappar hade även varit bra att ha i fönstret där det gick att ställa in vilka tillstånd som det genererades testfall för i modellen. I prototyperna blev användarna osäkra om deras val genomförts eller ej.

- I fönstret där täckningen av modellen ställdes in ville försökspersonen att det fanns något sätt att få information vad de olika alternativen innebar.

- En överblick av modellen hade varit bra i testgenererings- och testexekveringsvyn. I denna överblick skulle användarna kunna se vilka tillstånd i modellen som de genererade testfallen täckte respektive vilka tillstånd som ett specifikt testfall testade i modellen.

- En lista över testfallen i testsviten hade varit bra i testexekveringsvyn. Samtidigt skulle det ges möjlighet att välja bort ett eller flera testfall.

- När en modell raderas gavs ingen förfrågan om användaren verkligen ville ta bort modellen. Tyvärr kan något raderas av misstag ifall ingen förfrågan görs.

- Knapparna för att visa antingen fullständig informationslogg eller enbart felen i testexekveringsvyn kunde kombinerats till en knapp. Detta eftersom det endast gick att använda sig av den ena eller den andra beroende på vad som visades för tillfället.

- Knappkombinationer, t.ex. ”Shift+S” eller liknande, hade varit bra för att skifta mellan ritverktyg.

- Att kunna spara ett helt projekt, dvs. spara modell, testsvit samt testresultat i en fil, kändes förvirrande för en del försökspersoner eftersom de använde denna funktion för att spara sin modell.

- Textrutan som visade nuvarande algoritm- samt täckningsval lurade en del användare till att tro att det gick att editera valen via textrutan. En enkel lösning på detta hade varit att ta bort textrutan och bara ha en enkel textremsa som visade nuvarande valen.

2.7 Vidareutveckling av prototyp

Precis som vid föregående utvecklingsfas användes XP-programmering som arbetsmetodik. Det första som gjordes var att förbättringsförslagen ifrån föregående arbetsmoment överfördes till *user stories*. Baserat på de resultat som gavs av testningen valdes prototyp Alpha ut som grund för vidareutvecklingen. Detta val gjordes eftersom grundstommen, uppifrån-och-ner-perspektivet samt fliksystemet, redan var implementerade i prototyp Alpha. Den funktionalitet som fungerade bättre i Beta än i Alpha överfördes sedan till Alpha så att alla fördelarna hos de båda prototyperna kunde utnyttjas. Tyvärr kunde inte alla de önskade förbättringarna implementeras p.g.a. tidsbrist så en lista över de förbättringar som kom med samt de som inte kom med i den uppgraderade prototypen finns i bilaga E. De fem viktigaste *user stories* som implementerades i den nya prototypen var:

- När en testsvit öppnas visas det vilka testfall som ingår i testsviten. Det går dessutom att kolla på individuella testfall och se vad de testar samt välja om de ska ingå i testexekveringen.

- En ”OK”- och en ”Cancel”-knapp skapades i fönsterna där det gick att sätta algoritm och täckning. ”OK”-knappen sparar valen som gjorts och ”Cancel”-knappen återställer de val som var registrerade när fönstret öppnades.

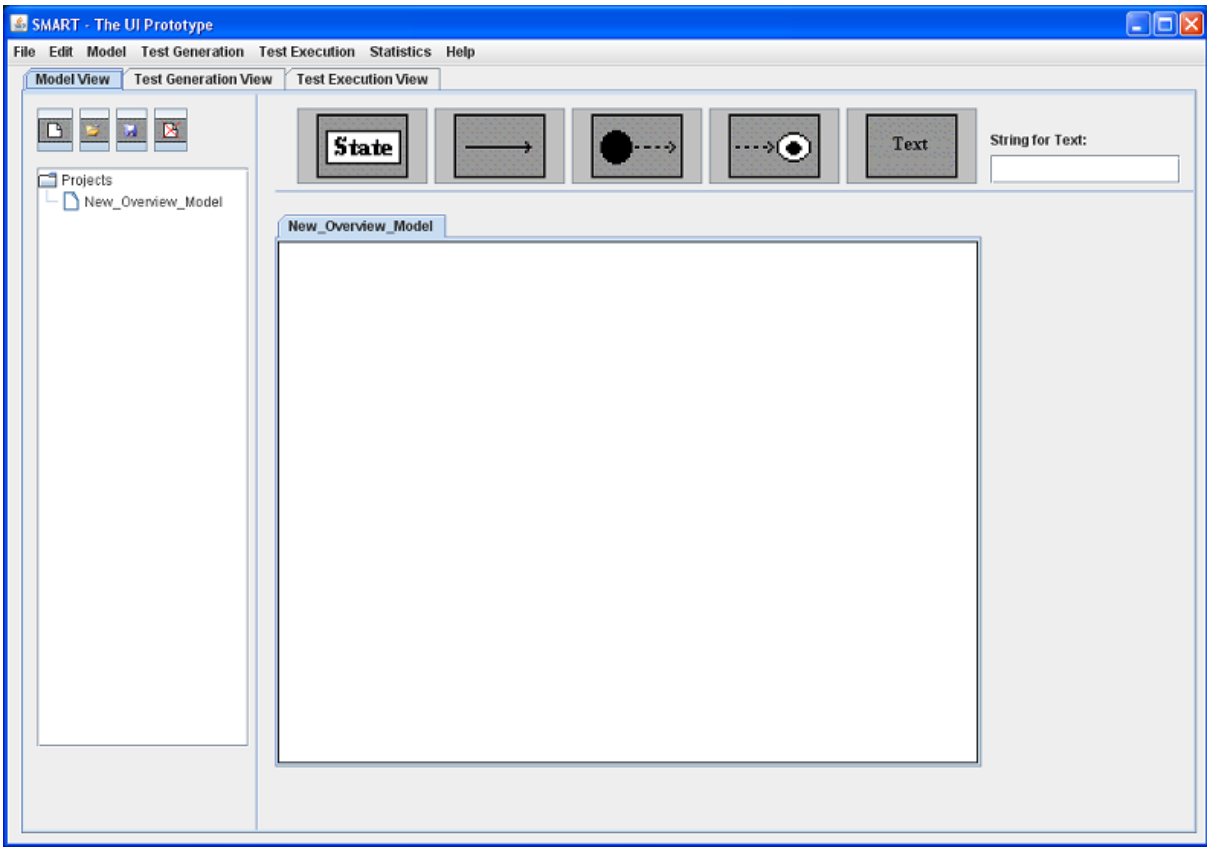
- Istället för att låta användaren hålla inne musknappen och placera ut ett objekt genom att släppa musknappen när det nått önskad position ändrades detta till att istället alltid visa en förhandsvisning av var objektet skulle hamna. Alltså visades objektet hela tiden och det var möjligt att flytta runt det. Det ”fastnade” inte på modellytan förrän användaren klickade på musknappen.

- Knappkombinationer för att byta modellverktyg lades till. Kombinationerna var enligt nedan:

    1. Tillståndsverktyget ”Shift+S” (state).
    2. Tillståndsändringsverktyget ”Shift+T” (transition).
    3. Startverktyget ”Shift+B” (beginning eftersom ”S” som i start redan var taget).
    4. Slutverktyget ”Shift+E” (end).
    5. Textverktyget ”Shift+T” (text).

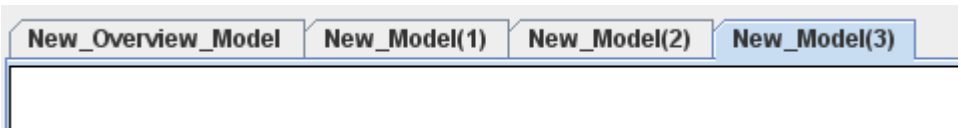
- En överblick av modellen med vilka tillstånd och tillståndsändringar som täcktes av de inställningarna som gjorts av användaren lades till i testgenereringsvyn.

En bild av modelleringsvyn kan ses i figur 2.7.1 nedan.



Figur 2.7.1: Modelleringsvyn i den vidareutvecklade prototypen.

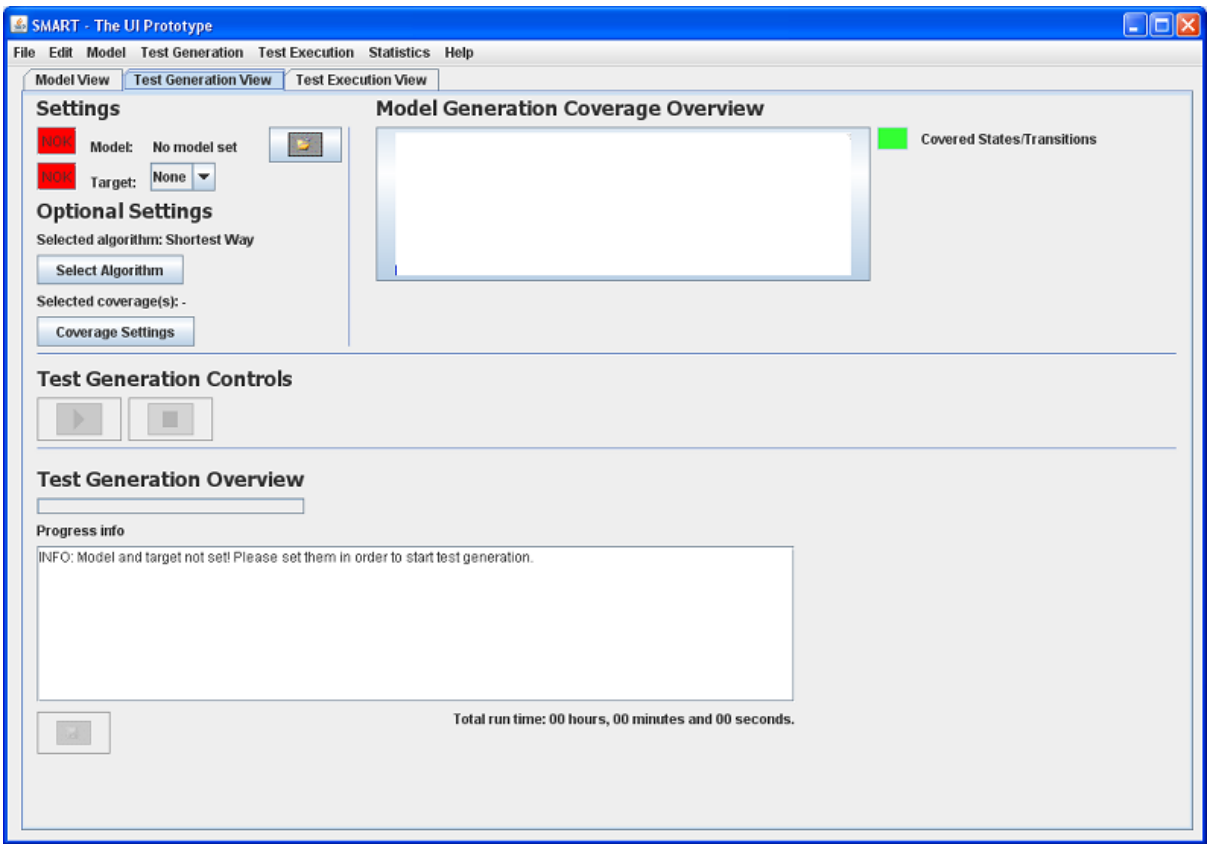
Det som kan ses är att ikonknapparna från prototyp Beta har kombinerats med konceptet att arbeta uppifrån-och-ner som användes i prototyp Alpha. Vidare används fliksystemet för att växla mellan vyerna, samma som det som användes i Alpha. Det fanns två sätt att växla mellan olika modeller. Antingen klickade användaren på en modell i trädstrukturen på vänster sida eller på en av flikarna i modellytan.



Figur 2.7.2: Flikarna för att växla mellan olika modeller i den vidareutvecklade prototypen.

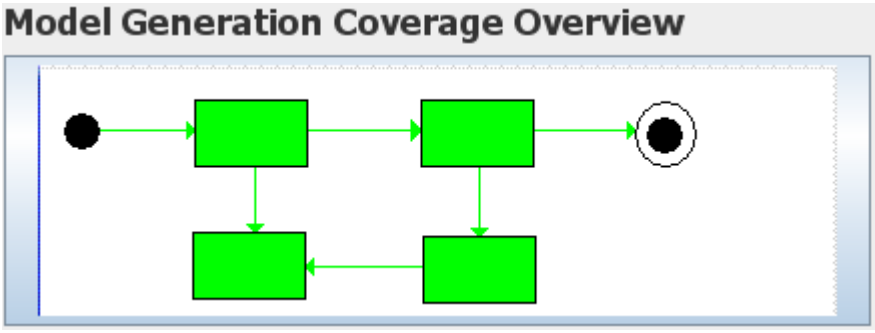
Under ett möte hos System Verification där en icke färdig vidareutvecklad prototyp visades upp var ikonerna för att skapa ny, öppna, spara och ta bort en modell lika stora som ikonerna för modelleringsverktygen. En diskussion ledde till att det bestämdes att ikonerna för att skapa ny, öppna, spara och ta bart en modell tog för stort fokus i denna vy eftersom de inte skulle användas lika mycket som modelleringsverktygen. Därför minskades storleken på dessa till 50 % av den ursprungliga så att användarens fokus hamnade på modelleringsverktygen. Precis som tidigare grönmarkerades bakgrunden på det verktyg som var aktivt så att användaren snabbt kunde avgöra vilket verktyg som var valt. Största skillnaden i denna vy gentemot tidigare prototyper var sättet att placera objekt. Istället för att hålla inne musknappen och flytta runt objektet tills önskad position nåtts så ritades istället en förhandsvisning av objektet ut och användaren behövde endast trycka för att låsa objektet till modellytan.

I testgenereringsvyn, som kan ses i figur 2.7.3, användes också grundprincipen att arbeta uppifrån-och-ner.



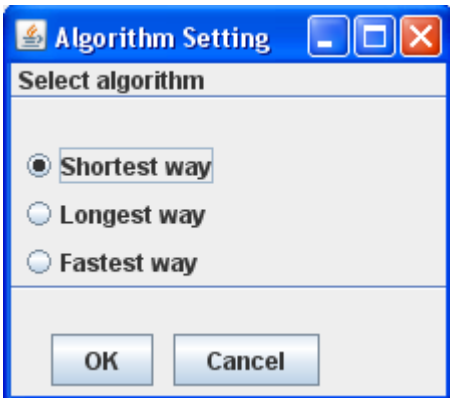
Figur 2.7.3: Testgenereringsvyn i den vidareutvecklade prototypen.

Fem av knapparna har fått ikoner i den här vyn. Två av dessa knappar var de för att spara eller öppna något som fick standardikonen för respektive funktion, dvs. en diskett för att spara samt en öppen mapp för att öppna. De tre sista knapparna som fick ikoner var de för att starta, pausa och stoppa testgenereringen. Ikonerna som användes i prototyp Beta återanvändes till dessa knappar eftersom de fungerade bra där. En annan sak som också bör nämnas är att starta och pausa kombinerades till en knapp där ikonen ändrades beroende på om testgenereringen var igång eller inte. Däremot skrotades ikonerna för att välja algoritm och täckning eftersom de kändes mindre självklara för användarna. Istället användes text på dessa knappar som beskrev vad de gjorde, precis som i Alpha. Statusrutorna som visade om en modell eller plattform hade satts togs från Alpha eftersom användaren genom ett snabbt ögonkast kunde avgöra om de satts eller inte. Största skillnad gentemot tidigare prototyper var att en översikt av modellen visades. I denna kunde användaren se vilka tillstånd och tillståndsändringar som skulle täckas av testfallen.



Figur 2.7.4: Ett exempel på hur överblicken av modellen kunde se ut i testgenereringsvyn.

En annan sak som också hade ändrats var fönsterna för att välja algoritm och täckning. Till skillnad från tidigare prototyper fanns det nu knappar för att acceptera eller neka valen som gjordes. Hur de nya fönsterna såg ut kan ses i figur 2.7.5 och 2.7.7.

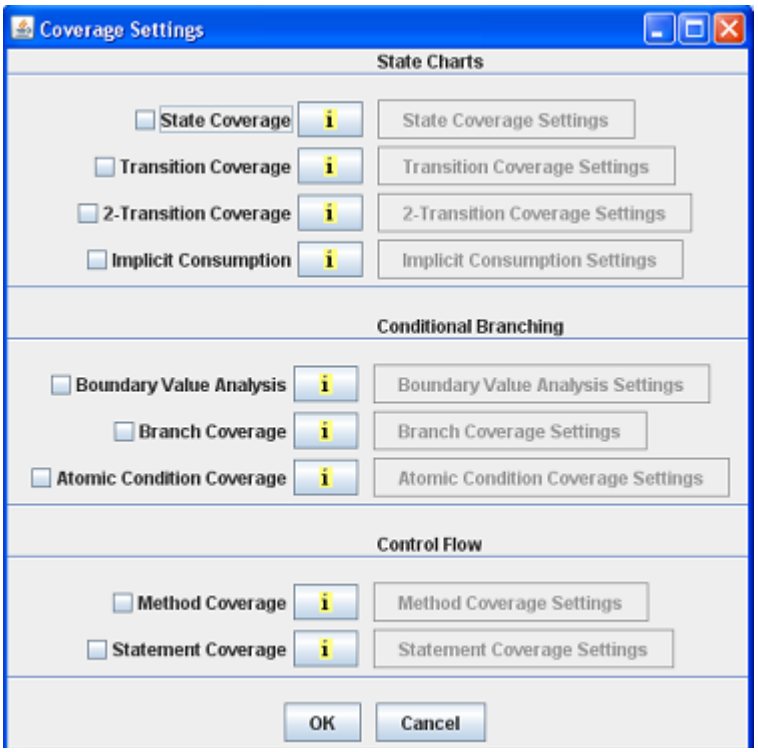


Figur 2.7.5: Fönstret för att välja algoritm i den vidareutvecklade prototypen.

Utöver knappar för att acceptera eller neka valen lades även knappar för att ge information om de olika täckningsalternativen till i täckningsfönstret. Dessa knappar fungerade på ett sådant sätt att om användaren la muspekaren ovanför knappen visades en beskrivning i tooltipet.

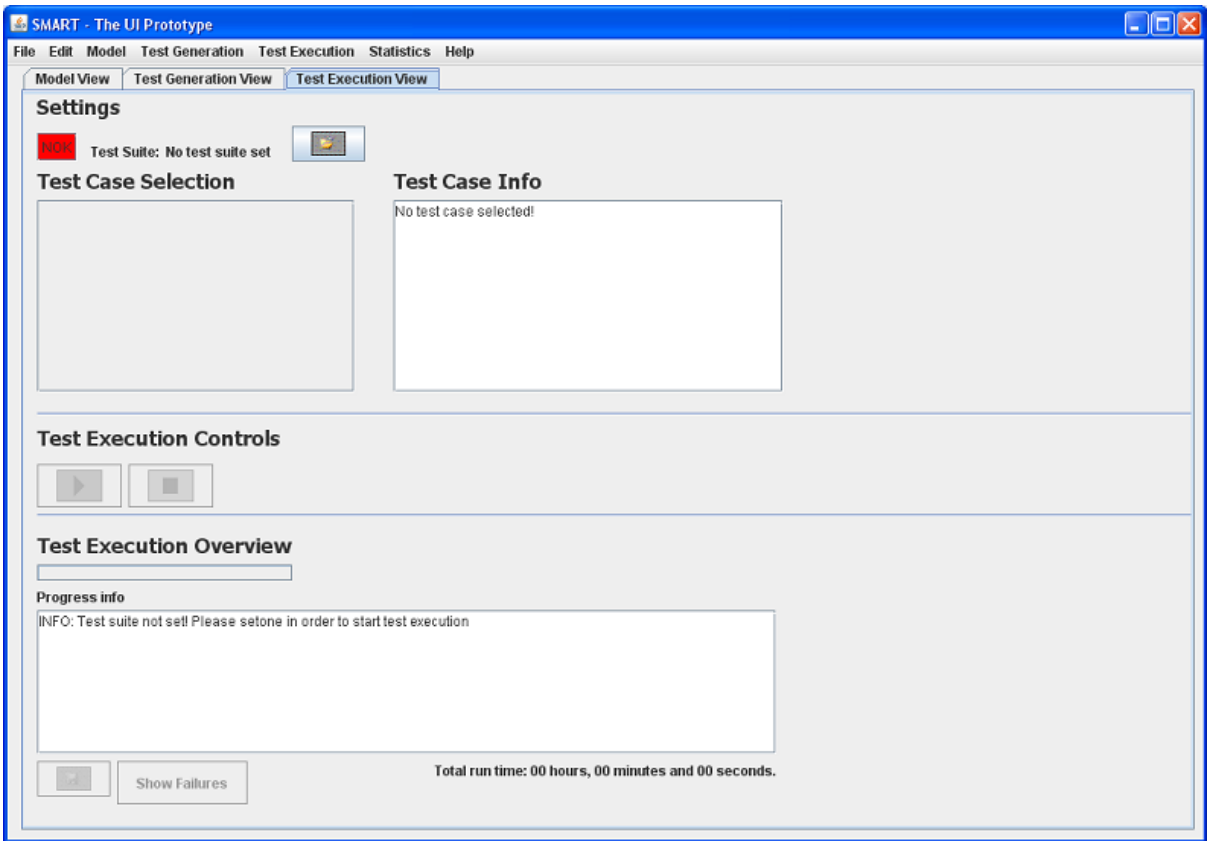
Figur 2.7.6: Tooltip med beskrivning av ett av täckningsalternativen.

Egentligen skulle tooltipet också visats när en användare klickade på en sådan knapp men någon sådan lösning kunde tyvärr inte hittas. Förutom detta lades det till knappar för att hantera avancerade inställningar för respektive täckningsalternativ. Dessa aktiverades först efter att ett täckningsalternativ valts.



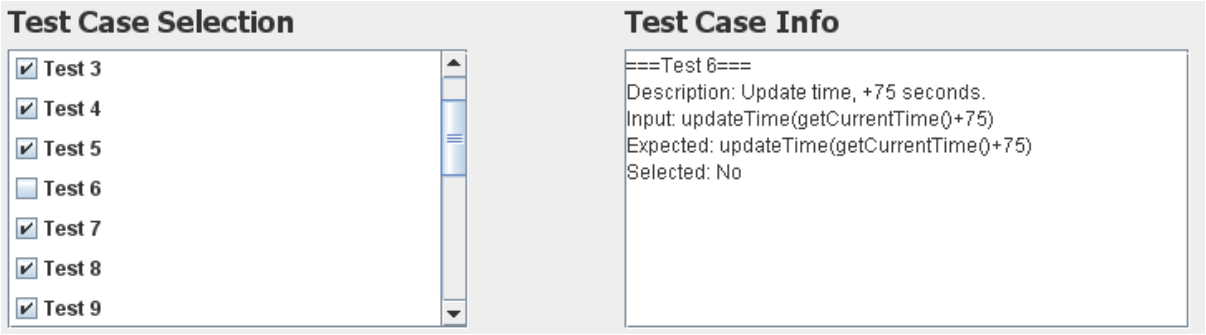
Figur 2.7.7: Fönstret för att välja täckning i den vidareutvecklade prototypen.

I den tredje och sista vyn, testexekveringsvyn, var mycket sig likt både från prototyp Alpha och från föregående vy, testgenereringsvyn.



Figur 2.7.8: Testexekveringsvyn i den vidareutvecklade prototypen.

I denna vy arbetade användaren uppifrån och ner precis som i de tre tidigare vyerna. Statusrutan för att visa om en testsvit var satt eller inte fanns även i den vidareutvecklade prototypen. Knapparna har använt samma ikoner som i testgenereringsvyn, dvs. standardikonerna för att öppna och spara samt ikonerna för att starta, pausa och stoppa exekveringen. En annan sak som också bör nämnas är att starta och pausa kombinerades till en knapp där ikonen ändrades beroende på om testexekveringen var igång eller inte. Den stora skillnaden mot tidigare prototyper var att det nu visades vilka testfall som ingick i en testsvit. Det var även möjligt att välja till eller bort ett testfall samt läsa mer om testfallet som markerats. Att välja till eller bort ett testfall gjordes genom att trycka på den kryssruta som var länkat till testfallet, se figur 2.7.9.



Figur 2.7.9: Testfallsöverblick i den vidareutvecklade prototypen.

2.8 Användartestning av vidareutvecklad prototyp

Efter att den vidareutvecklade prototypen hade nått ett stadium som kändes som en klar förbättring ifrån föregående versioner så genomfördes en andra testning. Precis som vid föregående testning tillhandahöll System Verification ett antal testpersoner. Tyvärr kunde endast tre av fem tillfrågade personer, samma som föregående testomgång, närvara. För att få lika mycket information vid denna testning som vid förra valdes två teknikstudenter från LTH ut för att fylla ut bortfallet. Uppgiftsblanketten ändrades från föregående testning på grund av att uppgifterna hade anpassats något till den nya prototypen. Dessutom hade en del av begränsningarna åtgärdats och en del andra begränsningar hade uppmärksammats under föregående testning. Uppgiftsblanketten för den andra testningen finns att beskåda i bilaga D. Tillvägagångssättet under själva testningen var exakt samma som vid föregående testning med enda skillnaden att ingen svarsenkät fylldes i efter att uppgifterna slutförts. Istället hölls en muntlig diskussion mellan testledaren och testpersonen där testledaren. Efter att all data från testningen analyserats skapades en lista över sådant som behövde förbättras. Syftet med andra testningen var först och främst att kontrollera att de åtgärder som gjorts fungerade på ett sätt som gjorde att användarupplevelsen förbättrades.

De tre försökspersonerna som kom från System Verification hade ett eller flera års erfarenhet av testning och de två försökspersonerna som var från LTH hade ingen erfarenhet av testning alls. Samtliga försökspersoner hade någon form av teknisk utbildning bakom sig, vilket kan ses i tabell 2.8.1 nedan.

Tabell 2.8.1: Beskrivning av försökspersonerna.

Försöksperson:	**Utbildning:**	**Antal års erfarenhet av testning:**	**Tidigare erfarenhet av modellbaserad testning:**
FP1	Interaktionsvetenskap	1-3 år	Nej
FP2	Teknisk-Fysik	4-7 år	Ja
FP3	Interaktionsvetenskap	1-3 år	Nej
FP4	Datateknik	0 år	Nej
FP5	Datateknik	0 år	Nej

Tabell 2.8.2: Data för försöksperson 1.

Person: FP1		
Uppgift:	**Tid:**	**Antal fel:**
Rita och spara en modell. Motsvarar uppgift 1 och 2 på uppgiftsblanketten.	3 min och 24 s	1
Generera testfall och spara som en testfallssvit. Motsvarar uppgift 3 och 4 på uppgiftsblanketten.	1 min och 32 s	0
Exekvera testfallen i en testfallssvit. Motsvarar uppgift 5 på uppgiftsblanketten.	1 min och 10 s	0

Felet som FP1 gjorde var:

1. Valde textverktyget men glömde att skriva en text i textfältet innan han klickade i modellytan. På grund av att inget textvärde angetts så skrevs ingen text ut.

Värdefulla kommentarer och svar från FP1 på frågorna som ställdes av testledaren:

1. En överblick av modellen i testgenereringsvyn var användbar och gav användaren snabbt information om vad i modellen som täcktes med de inställningar som gjorts.
2. En överblick av modellen hade varit användbar i testexekveringsvyn.
3. Istället för att ha två olika knappar för inställningarna för testgenerering hade det varit bra att ha en knapp som öppnade ett fönster som innehöll flikar för de olika inställningarna.
4. Gillade förhandsvisningen av objekten bättre än att hålla inne knappen och sen släppa när objektet nått rätt position.
5. Tyckte att det kvittade om det användes en eller två klick för att skapa en tillståndsändring.
6. När det klickas på ett objekt hade det varit bra om det gick att få upp extra valmöjligheter på ett annat ställe i eller vid sidan om modellvyn. Samtidigt skulle det objekt som det klickades på markeras.
7. Tyckte det var överflödigt med grön färg på verktygsknapparna. Det hade räckt med att muspekaren ändrades beroende på valt verktyg.
8. Knappkombinationerna för att ändra verktyg var positivt.
9. Skulle vilja att knappkombinationen skrevs ut i tooltipet på ett eller annat sätt.
10. Tyckte att informationsknapparna för de olika täckningsalternativen var överflödiga. Istället kunde tooltipet på dessa knappar läggas till på kryssrutorna.
11. Hade svårt att förstå att ”NOK” var förkortning för ”Not OK”.
12. Upplevde att förklaringen för hur stor del av modellen som täcktes, den gröna rutan med texten ”Covered States/Transitions”, var överflödig.
13. Önskade att det gavs mer information under testgenerering och exekvering, t.ex. var det var fel i en modell och exakt vad som skapades/exekverades.
14. Ville ha möjlighet att välja eller välja bort flera testfall på en och samma gång. Även möjlighet för att markera eller avmarkera alla testfallen med ett klick.
15. Upplevde att uppdelningen i tre flikar var logisk och bra. Onödigt att lägga för mycket information i samma flik genom att kombinera flera.
16. Ville ha en tydlig avgränsning mellan olika testexekveringar, t.ex. genom att skriva ut datum och tid för att identifiera när en exekvering startades.

Tabell 2.8.3: Data för försöksperson 2.

Person: FP2		
Uppgift:	**Tid:**	**Antal fel:**
Rita och spara en modell. Motsvarar uppgift 1 och 2 på uppgiftsblanketten.	4 min och 20 s	2
Generera testfall och spara som en testfallssvit. Motsvarar uppgift 3 och 4 på uppgiftsblanketten.	2 min och 47 s	0
Exekvera testfallen i en testfallssvit. Motsvarar uppgift 5 på uppgiftsblanketten.	2 min och 50 s	0

Felen som FP2 gjorde var:

1. Försökte dra ut objekten från knapparna.
2. Valde textverktyget men glömde att skriva en text i textfältet innan han klickade i modellytan. På grund av att inget textvärde angetts så skrevs ingen text ut.

Värdefulla kommentarer och svar från FP2 på frågorna som ställdes av testledaren:

1. Gillade förhandsvisningen av objekten bättre än att hålla inne knappen och sen släppa när objektet nått rätt position.
2. Skulle föredra om det var två klick för att lägga till en tillståndsförändring.
3. Ville ha möjlighet att markera eller avmarkera alla testfallen med ett klick.
4. Tyckte att det var irriterande att ett testfall avmarkerades om användaren klickade på texten för testfallet. Istället borde testfallet bara markeras och behålla samma status som tidigare.
5. Knappkombinationerna för att ändra verktyg var positivt.
6. Skulle vilja att knappkombinationen skrevs ut i tooltipet på ett eller annat sätt.
7. Upplevde att det var onödigt med start, paus och stopp i menyerna då de troligtvis aldrig skulle användas.
8. Skulle vilja att det fanns något sätt att slumpa vilka tillstånd och tillståndsförändringar som täcktes vid en testgenerering.
9. Tyckte att inställningstexten för täckning och algoritm var bra eftersom det inte krävdes att man gick in i respektive inställningsfönster för att se valet.
10. Ville att informationsloggen och resultatet i testexekveringsvyn skulle delas upp i två olika fönster. På så sätt behövdes inte knappen för att alternera mellan dessa.
11. Önskade att det fanns ett verktyg för att markera flera objekt. Detta skulle funka så att det drogs en rektangel runt de objekt som skulle markeras.
12. Tyckte att det var onödigt att muspekaren visades då objekten skulle ritas ut eftersom du hela tiden hade en förhandsvisning av objektet i fråga.

13. Tyckte att det var frustrerande att tillstånd fortfarande visades efter att muspekaren flyttats utanför modellytan.
14. Om användaren högerklickade på ett objekt skulle ytterligare inställningar dyka upp i modellytan.
15. Tyckte att texterna ”OK” och ”NOK” var lätta att förstå och gav bra information om att något var korrekt satt eller inte. Färgen för respektive, grön och röd, var också till nytta.

Anmärkning: FP2 missuppfattade uppgift 5 och valde att testfall 6, 11 och 18 skulle exekveras istället för alla utom just dessa.

Tabell 2.8.4: Data för försöksperson 3.

Person: FP3		
Uppgift:	**Tid:**	**Antal fel:**
Rita och spara en modell. Motsvarar uppgift 1 och 2 på uppgiftsblanketten.	4 min och 13 s	1
Generera testfall och spara som en testfallssvit. Motsvarar uppgift 3 och 4 på uppgiftsblanketten.	1 min och 57 s	0
Exekvera testfallen i en testfallssvit. Motsvarar uppgift 5 på uppgiftsblanketten.	1 min och 36 s	0

Felet som FP3 gjorde var:

1. Valde textverktyget men glömde att skriva en text i textfältet innan han klickade i modellytan. På grund av att inget textvärde angetts så skrevs ingen text ut.

Värdefulla kommentarer och svar från FP3 på frågorna som ställdes av testledaren:

1. En överblick av modellen i testgenereringsvyn var användbar och gav användaren snabbt information om vad i modellen som täcktes med de inställningar som gjorts.
2. En överblick av modellen hade varit användbar i testexekveringsvyn.
3. Skulle vilja att det gick att ändra namn på modellerna direkt i trädstrukturen.
4. Tyckte att det kvittade om det användes en eller två klick för att skapa en tillståndsändring.
5. När programmet stängs ner borde det frågas om osparat material ska sparas.
6. Ville ha möjlighet att välja eller välja bort flera testfall på en och samma gång. Även möjlighet för att markera eller avmarkera alla testfallen med ett klick.
7. Den kontinuerliga återkopplingen från informationsloggarna var bra.
8. Färgkodningen för att visa om en modell eller liknande satts korrekt för att generera/exekvera testfall var till stor hjälp för att snabbt avgöra om något var ok eller inte.

9. Tyckte att det var frustrerande att tillstånd fortfarande visades efter att muspekaren flyttats utanför modellytan.
10. Tyckte att ikonerna i programmet såg fula ut och skulle vilja att de var bättre gjorda. Samtidigt ville han att ikonerna skulle täcka upp hela knapparna.
11. Vid programstart första gången skulle en hjälpmeny eller kanske någon form av ”Tip of the day” kunna visats.

Tabell 2.8.5: Data för försöksperson 4.

Person: FP4		
Uppgift:	**Tid:**	**Antal fel:**
Rita och spara en modell. Motsvarar uppgift 1 och 2 på uppgiftsblanketten.	3 min och 22 s	1
Generera testfall och spara som en testfallssvit. Motsvarar uppgift 3 och 4 på uppgiftsblanketten.	2 min och 22 s	0
Exekvera testfallen i en testfallssvit. Motsvarar uppgift 5 på uppgiftsblanketten.	1 min och 52 s	0

Felet som FP4 gjorde var:

1. Försökte rita en tillståndsändring genom två musklickningar. Ingenting ritades ut då det krävs att musknappen hålls inne och sen släpps på en ny position.

Värdefulla kommentarer och svar från FP4 på frågorna som ställdes av testledaren:

1. En överblick av modellen i testgenereringsvyn var användbar och gav användaren snabbt information om vad i modellen som täcktes med de inställningar som gjorts.
2. En överblick av modellen hade varit användbar i testexekveringsvyn.
3. Trodde att han skulle föredra den nuvarande metoden för att rita objekt framför den föregående, dvs. förhandsvisning av objekt var att föredra framför att hålla inne musknappen och placera objekten.
4. Tyckte att det var mer logiskt att använda två musklickningar för att rita en tillståndsändring än att hålla inne musknappen.
5. Ville att muspekaren skulle ändra utseende om start- eller slutpunkt samt tillstånd var valt som verktyg.
6. Knappkombinationerna för att ändra verktyg var positivt.
7. Ville ha möjlighet att markera eller avmarkera alla testfallen med ett klick.
8. Upplevde att uppdelningen i tre flikar var logisk och bra. Onödigt att lägga för mycket information i samma flik genom att kombinera flera.
9. Tyckte att menyerna var ganska onödiga då han inte använde dem under testets gång.
10. Ville att informationsloggen och resultatet i testexekveringsvyn skulle delas upp i två olika fönster. På så sätt behövs inte knappen för att alternera mellan dessa.

11. Önskade att det fanns ett verktyg för att markera flera objekt. Detta skulle funka så att det drogs en rektangel runt de objekt som skulle markeras.
12. Undrade varför det inte gick att välja mer än en algoritm.
13. Tyckte att ikonerna i programmet såg fula ut och skulle vilja att de var bättre gjorda. Samtidigt ville han att ikonerna skulle täcka upp hela knapparna.
14. Skulle vilja att knappkombinationen skrevs ut i tooltipet på ett eller annat sätt.

Tabell 2.8.6: Data för försöksperson 5.

Person: FP5		
Uppgift:	**Tid:**	**Antal fel:**
Rita och spara en modell. Motsvarar uppgift 1 och 2 på uppgiftsblanketten.	4 min och 37 s	2
Generera testfall och spara som en testfallssvit. Motsvarar uppgift 3 och 4 på uppgiftsblanketten.	1 min och 41 s	0
Exekvera testfallen i en testfallssvit. Motsvarar uppgift 5 på uppgiftsblanketten.	1 min och 32 s	0

Felen som FP5 gjorde var:

1. Valde textverktyget men glömde att skriva en text i textfältet innan han klickade i modellytan. På grund av att inget textvärde angetts så skrevs ingen text ut.
2. Försökte rita en tillståndsändring genom två musklickningar. Ingenting ritades ut då det krävs att musknappen hålls inne och sen släpps på en ny position.

Värdefulla kommentarer och svar från FP5 på frågorna som ställdes av testledaren:

1. Ville ha möjlighet att se övergripande statistik på något sätt.
2. Den kontinuerliga återkopplingen från informationsloggarna var väldigt bra. Det i kombination med progressbarsen gjorde att användaren aldrig trodde att programmet hade hängt sig.
3. Ville att informationsfönsterna skulle ta mindre fokus och plats än vad de gör för tillfället.
4. Tyckte att det var mer logiskt att använda två musklickningar för att rita en tillståndsändring än att hålla inne musknappen.
5. Tyckte att det var frustrerande att tillstånd fortfarande visades efter att muspekaren flyttats utanför modellytan.
6. En överblick av modellen i testgenereringsvyn var användbar och gav användaren snabbt information om vad i modellen som täcktes med de inställningar som gjorts.
7. En överblick av modellen hade varit användbar i testexekveringsvyn.

8. Färgkodningen för att visa om en modell eller liknande satts korrekt för att generera/exekvera testfall var till stor hjälp för att snabbt avgöra om något var ok eller inte.
9. Knappkombinationerna för att ändra verktyg var positivt.
10. Ville ha möjlighet att markera eller avmarkera alla testfallen med ett klick.
11. Upplevde att uppdelningen i tre flikar var logisk och bra. Onödigt att lägga för mycket information i samma flik genom att kombinera flera.

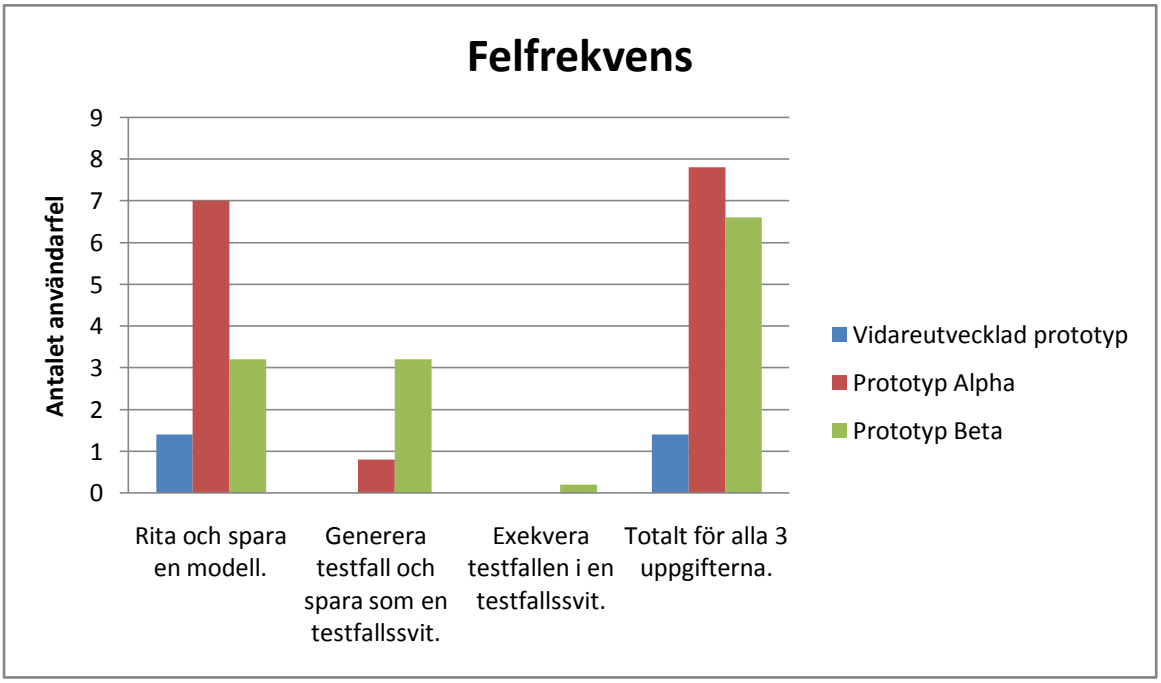
Tabell 2.8.7: Genomsnittlig data per person från utvärdering av uppdaterad prototyp.

Genomsnittlig data per person		
Uppgift:	**Tid:**	**Antal fel:**
Rita och spara en modell. Motsvarar uppgift 1 och 2 på uppgiftsblanketten.	3 min och 59,2 s	1,4
Generera testfall och spara som en testfallssvit. Motsvarar uppgift 3 och 4 på uppgiftsblanketten.	2 min och 3,8 s	0
Exekvera testfallen i en testfallssvit. Motsvarar uppgift 5 på uppgiftsblanketten.	1 min och 48 s	0

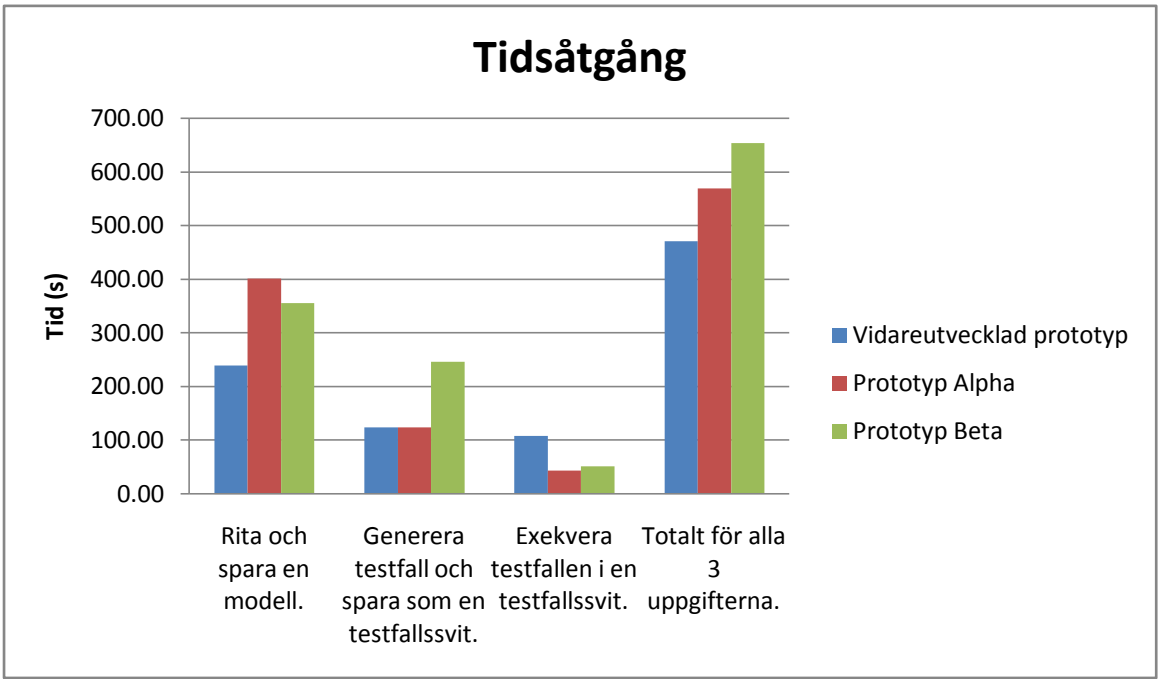
Tabell 2.8.8: Genomsnittlig data per person från tidigare utvärdering av prototyperna.

Genomsnittlig data per person	Prototyp: Alpha	
Uppgift:	**Tid:**	**Antal fel:**
Rita och spara en modell. Motsvarar uppgift 1 och 2 på uppgiftsblanketten.	6 min och 41,2 s	7
Generera testfall och spara som en testfallssvit. Motsvarar uppgift 3 och 4 på uppgiftsblanketten.	2 min och 4 s	0,8
Exekvera testfallen i en testfallssvit. Motsvarar uppgift 5 på uppgiftsblanketten.	0 min och 44 s	0
Genomsnittlig data per person	Prototyp: Beta	
Uppgift:	**Tid:**	**Antal fel:**
Rita och spara en modell. Motsvarar uppgift 1 och 2 på uppgiftsblanketten.	5 min och 55 s	3,2
Generera testfall och spara som en testfallssvit. Motsvarar uppgift 3 och 4 på uppgiftsblanketten.	4 min och 6,6 s	3,2
Exekvera testfallen i en testfallssvit. Motsvarar uppgift 5 på uppgiftsblanketten.	0 min och 52 s	0,2

Antalet användarfel i den uppgraderade prototypen var färre än i de tidigare prototyperna. Samtliga tre huvudfunktioner närmade sig målet att antalet användarfel skulle vara nära noll. Tiden för att rita och spara en modell minskade ganska markant. Detta kan dock ha berott på att testarna från System Verification hade använt programmet tidigare och därför hade lärt sig grunderna. Även studenterna använde mindre tid för att rita och spara en modell än vad testarna använde vid första testning. En annan sak som kan ha inverkat på resultatet var att en annan modell ritades än vid föregående test. Det skulle kunna vara så att den nya modellen var lättare att rita. Tiden försökspersonerna använde för att generera testfall var i princip samma nu som den var i prototyp Alpha. Dock var detta ingen direkt överraskning då testgenereringsvyn för den nya prototypen och Alpha är ganska snarlika till utseendet. Något som däremot kunde ses var en ökning av tiden för att exekvera testfallen. Detta kan ha berott på att det nu fanns fler inställningar som behövde göras än vid föregående testning.



Figur 2.8.1: Det totala antalet användarfel per prototyp samt antalet användarfel per uppgift.



Figur 2.8.2: Den totala arbetstiden per prototyp samt arbetstid per uppgift.

2.8.1 Slutsatser från andra testningen

En lista över de positiva och mindre positiva detaljerna hos den nya prototypen finns nedan. Listan över de mindre positiva detaljerna användes senare som checklista för att veta vad som behövde ändras till nästa version av prototypen.

Sådant som var positivt hos den nya prototypen och som inte registrerades under föregående utvärdering var:

- Kontinuerlig återkoppling, i form av progressbar och informationslogg, när testfall genereras och exekveras.
- Överblick av modellen i testgenereringsvyn, lätt att se vilka tillstånd och tillståndsförändringar som täcktes av testfallen som skulle genereras beroende på inställningarna som gjorts.
- Färgkodningen hjälpte till att avgöra om en inställning gjorts eller inte. Likaså gavs det snabb återkoppling om vilket verktyg som var valt.
- Knappkombinationer för att växla mellan verktyg är bra för avancerade användare för att kunna arbeta effektivare.
- Uppdelning av funktionaliteten i tre flikar, gjorde att användaren kunde koncentrera sig på en uppgift i taget.
- Förhandsvisning av var ett objekt skulle hamna var bättre än att hålla inne musknappen och sen flytta på det tills önskad position nåtts.

Negativt som bör förbättras inför nästa version samt förslag på ändringar:

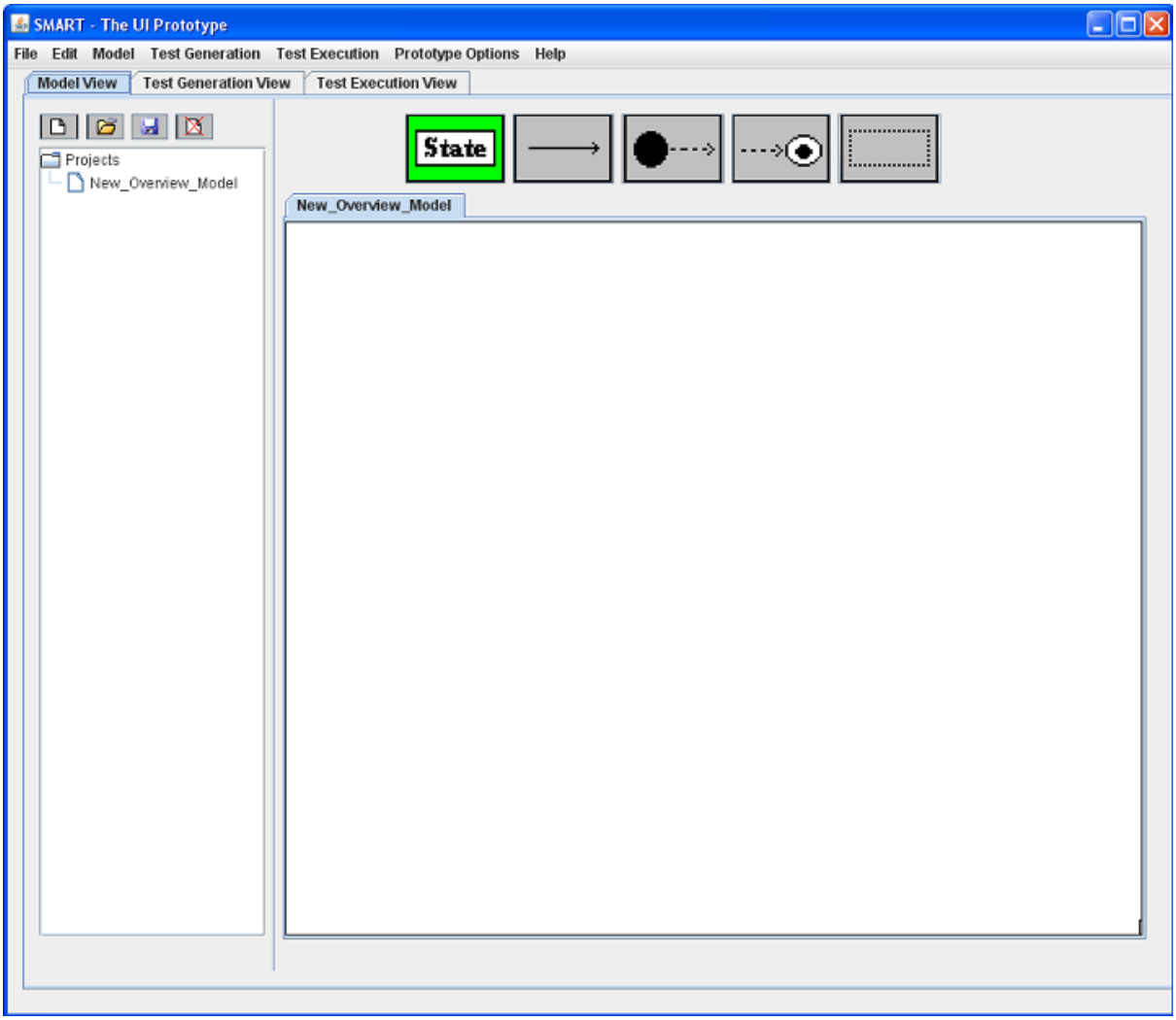
- En överblick av modellen i testexekveringsvyn hade också varit bra att ha.
- Ge möjlighet att rita en tillståndsförändring genom att klicka först på det initiella och sen på det avslutande objektet. På så sätt skulle en tillståndsändring ritas mellan dessa båda objekt där pilen riktades mot det avslutande objektet.
- Knappkombinationerna skulle kunna skrivas ut i tooltipet på knapparna för verktygen.
- Skriva ut datum och tid för när olika testgenereringar och exekveringar startas så att det går att skilja på dem ifall flera körts.
- Att klicka på texten för ett testfall innebar att testfallet valdes bort eller valdes beroende på tidigare inställning. Det skulle enbart vara möjligt att ändra inställningen genom att klicka på kryssrutan men så var inte fallet.
- Det hade varit enklare för användarna om textloggen och resultatet från testexekveringen hade delats upp i två olika fönster.
- Ett verktyg för att markera flera objekt hade varit bra. Förslagsvis genom att markera objekten som ska flyttas med en rektangel. Objekten skulle sen på något sätt visa att de markerats genom någon form av färgkodning.
- Muspekaren var onödig och ibland rent av irriterande när ett start-, slut- eller tillståndsobjekt skulle ritas ut eftersom det redan visades en förhandsvisning av var objektet skulle hamna.
- Om det finns osparat material när programmet ska stängas ner ska programmet fråga om det ska sparas.
- Vid den allra första starten av programmet skulle en hjälpmeny eller Tip-of-the-day-ruta varit uppskattat så att nya användare kunde fått en snabb genomgång av programmet.
- Borde vara möjligt att välja mer än en algoritm på samma gång.
- Försökspersonerna skulle vilja att det fanns möjlighet att slumpmässigt göra inställningarna för testgenerering.
- Infoknapparna som skulle ge information om vad de olika täckningsalternativen innebar fungerade inte särskilt bra.

2.9 Utveckling av slutgiltig prototyp

Arbetsprocessen vid sista utvecklingsfasen var exakt samma som vid föregående vidareutveckling. Extreme Programming användes och förslagen på förbättringar som tagits fram under andra testningen låg som grund för denna omgångs *user stories*. Tyvärr kunde inte alla de önskade förbättringar genomfördas p.g.a. tidsbrist eller tekniska begränsningar. En lista över de förbättringar som lades till och de som inte kunde realiseras i den slutgiltiga prototypen ses i bilaga F. De sex viktigaste förbättringar som implementerades i den slutgiltiga prototypen var:

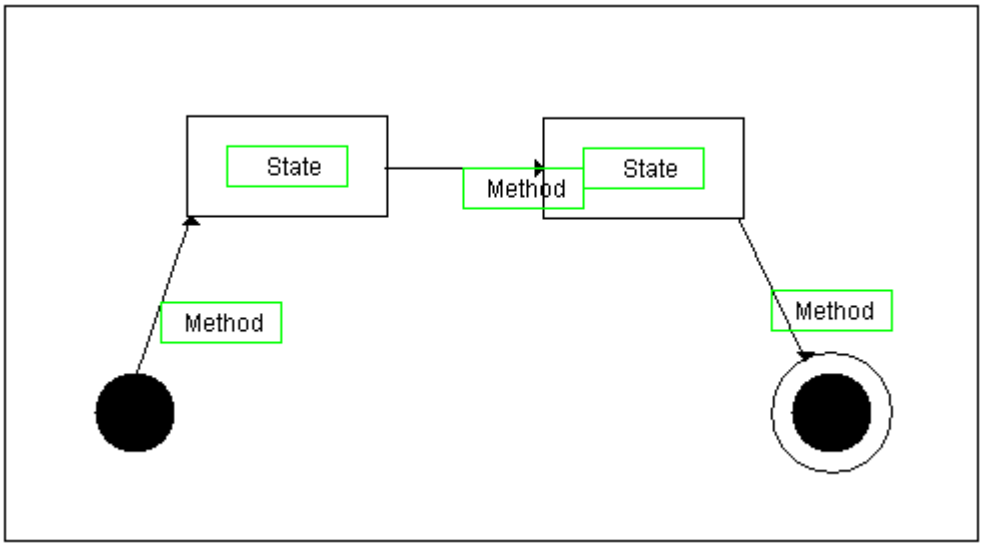
- Knappar för att markera eller avmarkera alla testfall lades till.
- Istället för att sätta ut text i efterhand skrivs det ut text direkt efter att ett tillstånd eller en tillståndsförändring skapats. Runt texten finns en grön rektangel och om användaren klickar innanför dess ram ska det vara möjligt att ändra texten. Att användaren var tvungen att klicka innanför ett markerat område skulle tydliggöra för användaren var hon skulle klicka för att ändra texten samt var hon skulle klicka för att markera objektet. Klickades det utanför textrutan skulle istället objektet markerats. Dock går det inte att ändra texten i prototypen p.g.a. tekniska begränsningar. Textverktyget som användes vid tidigare testningar skrotades med andra ord.
- Lagt till ett verktyg för att markera flera objekt.
- En knapp för att slumpmässigt ställa in täckningen av en modell har blivit tillagd i testgenereringsvyn.
- En överblick av modellen har lagts till i exekveringsvyn. Varje testfall kommer att visa vilka tillstånd och tillståndsändringar i modellen som testas.
- Muspekarna blev ändrade för modellverktygen. När ett start-, slut- eller tillståndsobjekt ska ritas ut visas ingen pekare alls och när en tillståndsändring ska ritas visas ett kors där mittpunkten är tillståndsändringens start- respektive slutpunkt.

Det slutgiltiga resultatet för modellvyn blev enligt nedan.



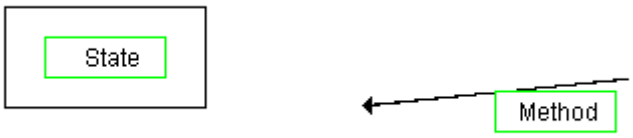
Figur 2.9.1: Modelleringsvyn i den slutgiltiga prototypen.

En skillnad mot tidigare prototyp var att textverktyget togs bort och istället lades ett verktyg för att markera flera objekt till. Markeringsverktyget fungerade på så sätt att en yta, i form av en rektangel, ritades upp och alla objekt som fanns inom denna rektangel markerades.



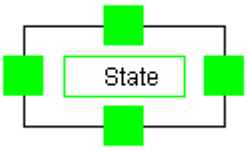
Figur 2.9.2: Rektangeln som används för att markera flera objekt.

Utöver detta uppdaterades ikonerna och knapparna så att de såg snyggare ut eftersom en del av försökspersonerna störtes av deras mindre snygga utseende. Den största ändringen gällde hur tillstånd och tillståndändringar ritades ut. Till skillnad från tidigare så skrevs nu en text med en grön rektangel runtom sig ut samtidigt som det ritades ut ett tillstånd eller en tillståndsändring. Tanken var, vilket tyvärr inte gick att genomföra med den tid som var tillgänglig, att om användaren skulle kunna editera texten om hon klickade innanför den gröna rektangeln där texten stod. Den gröna rektangeln skulle tydliggöra för användaren var i tillståndet eller tillståndsändringen som det skulle klickas för att göra det möjligt för användaren att ändra texten.



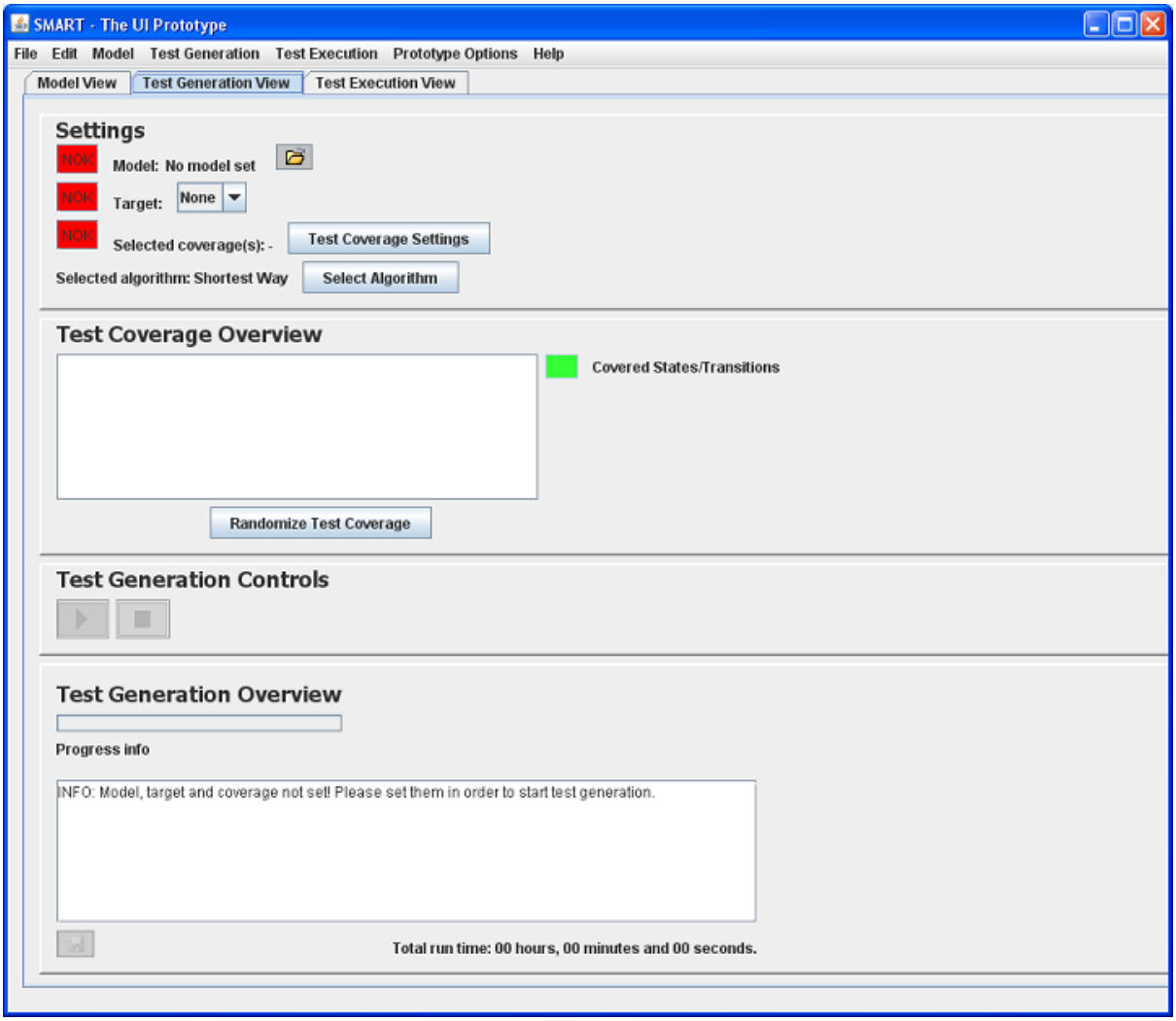
Figur 2.9.3: Tillstånd och tillståndsändring som de ritas ut i den slutgiltiga prototypen.

För att visa hur ett tillstånd skulle se ut efter att det markerats skapades en möjlighet att ändra sättet tillstånd ritades ut. Nedan visas hur ett tillstånd skulle se ut ifall det markerades. Tanken var att om användaren klickade på någon av de gröna kvadraterna så skulle ett nytt tillstånd skapas i det väderstrecket med en tillståndsändring ifrån det markerade tillståndet.



Figur 2.9.4: Hur ett tillstånd skulle se ut efter att det markerats i den slutgiltiga prototypen.

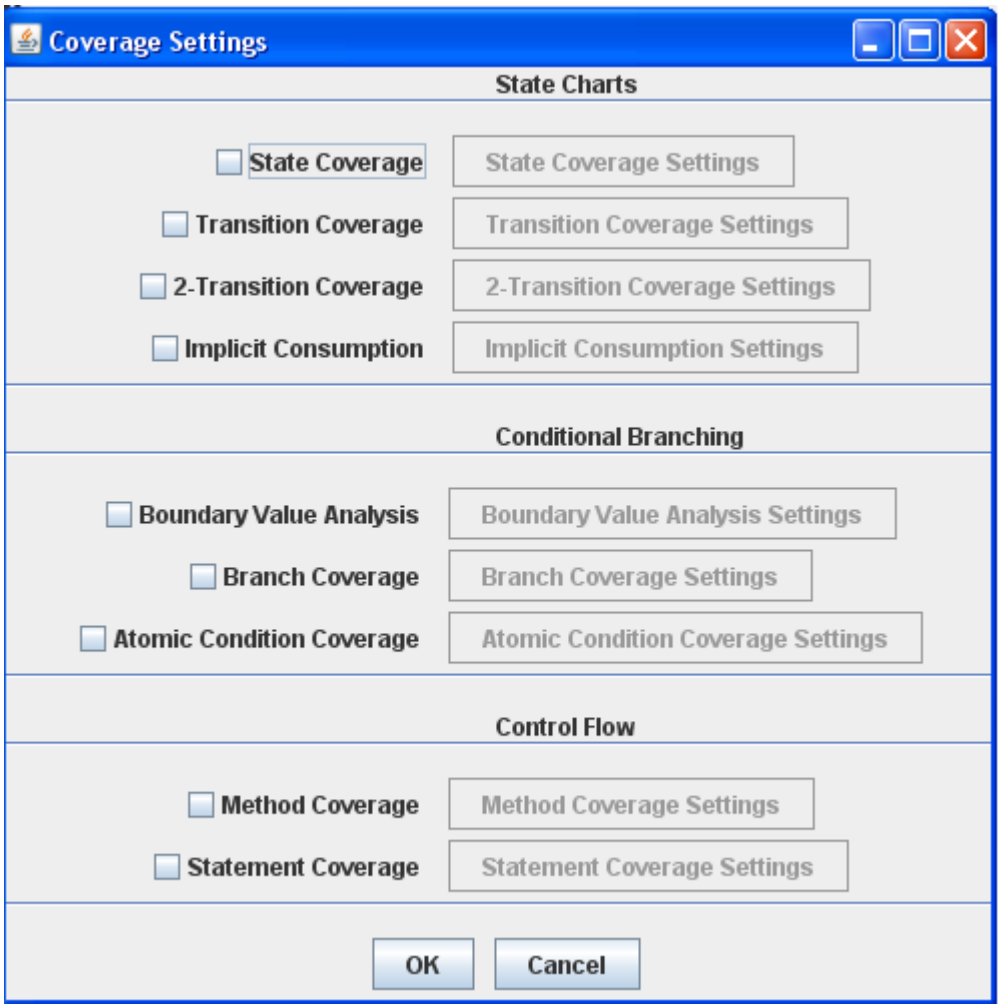
Vyn för testgenerering ändrades inte särskilt mycket ifrån föregående prototyp. En bild av denna vy kan ses i figur 2.9.5.



Figur 2.9.5: Testgenereringsvyn i den slutgiltiga prototypen.

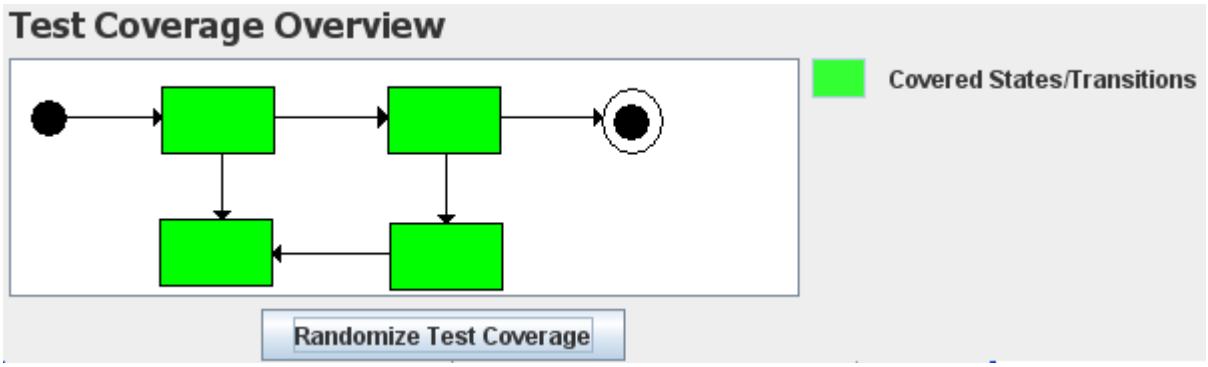
En av ändringarna som gjordes var att även täckningsinställningen fick en statusruta som visade om täckningen var konfigurerad korrekt eller inte. Eftersom det gick att välja bort alla täckningsalternativen måste användaren få någon form av återkoppling om att täckningsinställningen ej var korrekt satt. I fönstret för att ställa in testfallens täckning togs knapparna som visade information om varje täckningsalternativ bort. Istället lades ett tooltip

till på kryssrutorna som visade en beskrivande text om användaren höll muspekaren ovanför kryssrutan eller texten till kryssrutan.



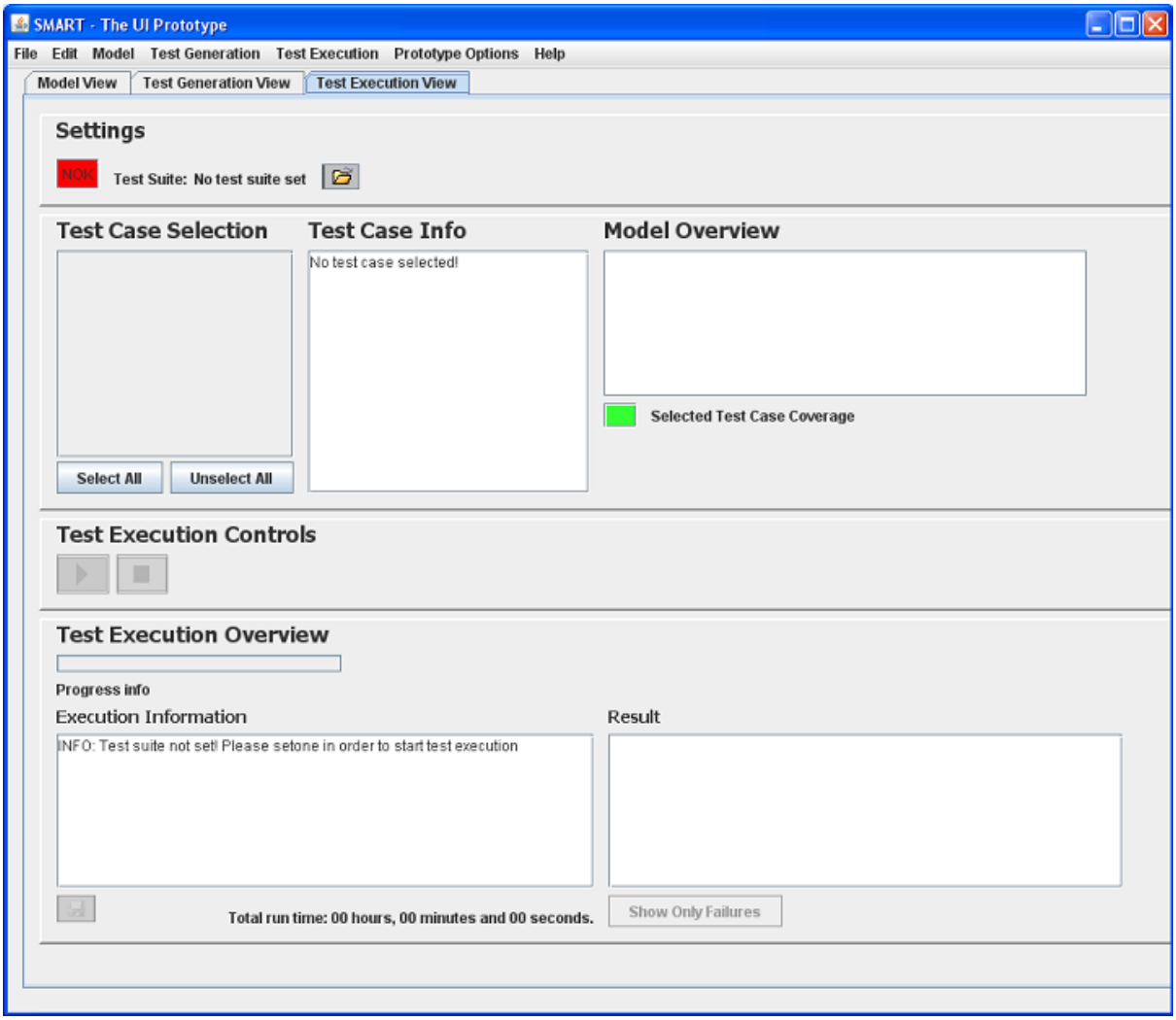
Figur 2.9.6: Det nya fönstret för att ställa in testfallens täckning.

Förutom ovanstående ändringar lades en knapp till för att slumpmässigt ställa in vilka tillstånd och tillståndsändringar som täcktes av testfallen som skulle genereras. Knappen sattes precis under överblicken av modellen så att återkopplingen över vad som täcktes av testerna var nära till hands. Funktionen i sig kan vara bra ifall en testare vill testa slumpmässigt.



Figur 2.9.7: Knappen för att slumpmässigt ställa in vilka tillstånd och tillståndsändringar som täcktes av testfallen.

Vyn som det ändrades mest i var testexekveringsvyn. De ändringar som märks mest är att det lagts till en överblick av modellen samt att informationsloggen delats upp i ett informationsfönster samt ett resultatfönster. Överblicken av modellen fungerade på exakt samma sätt som i testgenereringsvyn med den enda skillnaden att denna överblick visade vilka tillstånd och tillståndsändringar som ett specifikt testfall testade. Informationsloggen ändrades så att informationsfönstret gav användaren återkoppling om allt som skedde i vyn och att resultatfönstret presenterade resultatet av testexekveringen.



Figur 2.9.8: Testexekveringsvyn i den slutgiltiga prototypen.

3 Diskussion och slutsatser

Slutsatserna som drogs från utvärderingarna av befintliga test- och modelleringsprogram användes som grund för de initiella designerna. Detta gjorde att mindre bra designval som fanns hos dessa kunde undvikas och samtidigt kunde de designval som fungerade bra utnyttjas. Om mer tid hade funnits hade det kanske varit utav intresse att utvärdera fler program än vad det gjordes för att hitta andra smarta lösningar. En annan sak som också hade varit utav intresse hade varit att jämföra användbarheten hos min prototyp med de program som utvärderades. Tyvärr fanns det inte tid nog för att göra en sådan jämförelse.

Med hjälp av de två prototyperna som utvecklades kunde sedan en testomgång genomföras för att kontrollera vilka designprinciper som fungerade bäst. Det visade sig att testarna föredrog ett uppifrån-och-ner-perspektiv framför ett från-vänster-till-höger-perspektiv. Ett fliksystem för att växla mellan vyer fungerade bättre än ett system som använde sig utav knappar. Något som däremot inte var självklart var om användarna ville ha knapparna med ikoner eller text. I en del fall tyckte testarna att det kändes berättigat att använda sig utav ikoner men inte alltid. Slutsatsen blev att ikoner endast skulle användas där det gick att göra dem självförklarande. Givetvis skulle det varit möjligt att lägga till tooltip för de ikoner som inte var självklara. Det beslutades istället att text skulle användas på de knapparna som det inte gick att göra bra ikoner för eftersom texten var det som skulle visats i tooltipet. Andra positiva slutsatser som kunde dras ur den första testomgången var att färgkodning för att markera valt verktyg och vilka inställningar som gjorts eller inte gjorts fungerade bra. Att det kontinuerligt gavs återkoppling med hjälp av progressbars och informativ text gjorde att användaren alltid kände sig säker.

Under testningen upptäcktes också en mängd saker som behövde åtgärdas. De flesta sakerna åtgärdades under andra implementeringsfasen och för att säkerställa att de ändringar som gjorts var till användargränssnittets bästa utfördes en andra testomgång. Under denna testomgång märktes det att användargränssnittet hade förbättrats eftersom antalet användarfel och tiderna för att utföra uppgifterna minskade. Tiderna och frekvensen användarfel närmade sig de mål som satts upp på användbarheten. Sådant som var extra positivt med den vidareutvecklade prototypen var att användaren kunde få en överblick av modellen när hon skulle generera testfall så att det enkelt gick att avgöra om de tillstånds som önskades testas verkligen testades. Knappkombinationer för att ändra ritverktyg fungerade också bra eftersom de ökade användarnas produktivitet efter att de lärt sig dem. En annan sak som fungerade bra var att användaren fick möjlighet att välja exakt vilka testfall som skulle exekveras eller inte. En positiv förändring var att de objekt som skulle ritas ut förhandsvisades så att användaren alltid visste exakt var de skulle placeras. I tidigare prototyper var användarna ibland osäkra på var objekten skulle hamna.

Liksom vid föregående testomgång togs det även denna gång fram en lista på sådant som behövde fixas och merparten av dessa åtgärdades under en sista implementeringsomgång. Tyvärr kunde inte alla funktioner och designprinciper som önskades implementeras p.g.a. tekniska begränsningar samt tidsbrist. Sådant som skulle varit med om inga begränsningar funnits var möjligheten att markera ett objekt och sedan snabbt länka ett nytt tillstånd till det

eller att kopiera, eller ta bort ett markerat objekt. Det skulle också varit bra om resultat från testexekvering visades med färgkodning, t.ex. så skulle de testfall som inte godkändes visats med röd text medan de som godkändes skulle fått en grön text. En annan sak som också hade varit bra var om det vore möjligt att se miniatyrer av modellerna i trädstrukturen så användaren lätt kunde välja den modell som hon ville arbeta med.

Resultaten från testningarna av prototyperna visade att försökspersonerna upplevde saker som tydlighet, återkoppling, sortering och möjlighet att effektivisera arbetet som viktiga för ett användargränssnitt till ett modellbaserat testverktyg. För att tillgodose dessa behov kan nedanstående lista med riktlinjer vara utav hjälp. Riktlinjerna är baserade på resultaten som gavs vid testningarna. Även prototypen kan användas som underlag vid skapandet av ett modellbaserat testverktyg.

- Använd ikoner där det går att göra dem självförklarande. Det måste finnas någon referens från den riktiga världen, en standardikon eller ett sätt att visa det som ska skapas för att rättfärdiga att en ikon används. Om så inte var fallet skulle istället en knapp med text användas.
- Även om ikonerna ska vara självklara för användaren är det alltid bra att ge möjlighet att läsa tooltip. I prototypen som utvecklades skrevs t.ex. knappkombinationerna för ritverktygen ut så att användaren kunde lära sig dessa.
- Ge användaren en förhandsvisning av var objekten som ska skapas hamnar. Annars måste användaren flytta objektet i efterhand ifall det inte placerades exakt där det önskades.
- Tydliggör för användaren mellan vilka tillstånd som en tillståndsändring kopplas och åt vilket håll som tillståndsändringen sker.
- Använd färgkodning för att markera vilket verktyg som är valt för tillfället. Färgkodning kan också användas för att signalera vilka funktioner som inte går att använda för tillfället och för att visa vilka inställningar som gjorts.
- Ge kontinuerligt återkoppling till användaren i form av progressbars och text som skrivs ut när något genereras eller exekveras.
- Sortera funktioner på ett sådant sätt att användaren lätt kan hitta det hon vill använda.
- Skapa knappkombinationer för sådana funktioner som ofta används så att produktiviteten ökar.
- Låt användaren få möjlighet att se en överblick av modellen när testfall skapas och exekveras. Det blir lättare att skaffa sig en uppfattning om vad som testas och till vilka tillstånd det genereras testfall för.
- Använd filter för att öppna och spara filer så att endast de viktiga filerna visas och så att filer sparas med rätt filändelse.
- Ge möjlighet för användaren att välja bort eller till alla testfall i en testsvit. Det ska också vara möjligt att välja bort eller till enstaka testfall.

- Det ska vara möjligt att pausa eller stoppa en testgenerering eller exekvering så knappar för att göra detta måste finnas.
- Om ett tillstånd markeras ska en funktion som gör det möjligt att snabbt skapa en tillståndsändring och ett tillstånd ifrån det markerade tillståndet visas.
- Visa miniatyrer av de modellerna som finns i trädstrukturen så att användaren kan avgöra vilken som är vilken.
- Ge grafiskt information om vilka testfall som inte godkändes samt vilka som godkändes t.ex. genom att använda röda ikoner eller text för de som godkändes respektive grönt för de som godkändes.
- Dela upp modellritandet, testgenereringen och testexekveringen i egna flikar eller fönster så att inte alla tre funktionerna visas på samma gång.

I utvärderingen av befintliga testverktygs gränssnitt upptäcktes både bra och mindre bra detaljer. Det som fungerade bra var att ikonerna i programmen ofta hämtade inspiration från den riktiga världen vilket gjorde att användarna lätt kunde lista ut knappens funktion. Fanns det ingen bra bild som kunde hämtas därifrån användes istället det som skapades av funktionen som ikon. Normans konceptuella modeller användes ofta som underlag för ikonerna. Om en ikon var svår att förstå användes tooltip så att användaren kunde läsa vad knappens funktion var. Funktioner som för tillfället inte gick att använda gråmarkerades så att användaren lätt kunde se att de var omöjliga att använda. Något annat som också var positivt var att det verktyg som för tillfället var valt visades tydligt eftersom bakgrundsfärgen på knappen ändrades till en annan än de övrigas. Funktioner som hanterade liknande saker sorterades ofta tillsammans så att användaren lätt kunde avgöra var en specifik funktion skulle finnas. Antalet inställningar och funktioner som kunde göras från standardfönstret hos ett program begränsades tack vare att avancerade inställningar gömdes i andra fönster. En sista sak som också uppmärksammades var att det kontinuerligt gavs återkoppling. Detta gjorde att användaren hela tiden visste vad som ägde rum. För att se skillnad på sådant som gick fel och sådant som inte gick fel användes olika färger. Rött indikerade att något gått snett och grönt i sin tur visade att något blivit rätt, vilket motsvarar Normans term självinstruerande.

Det som fungerade mindre bra var att en del ikoner var alldeles för lika varandra vilket gjorde det svårt för användaren att förstå skillnaden i funktionalitet. I en del program var det även svårt att hitta en specifik funktion eftersom de hade sorterats på ett mindre bra sätt i menyerna. Ifall ett tillstånd eller annat objekt markerades visades ett antal funktioner upp i modelleringsytan. En del av dessa funktioner var bra medan merparten kändes överflödiga. Antalet funktioner som visas när ett objekt markeras borde istället minimeras så att användaren inte ges onödigt många alternativ. Det blev också svårt att använda en specifik funktion eftersom många funktioner presenterades på en väldigt liten yta. Efter att ett objekt skapats valdes det nyss valda verktyget bort. Ville användaren skapa ett till likadant objekt blev hon tvungen att välja om verktyget. Istället borde verktyget fortfarande vara valt efter att det använts. Något annat som inte heller fungerade särskilt bra var att fokuset inte byttes till det som skapades. Skapade användaren t.ex. en ny modell flyttades inte fokuset hos

programmet till den nya modellytan utan istället var fokuset kvar på föregående modellyta. En viktig detalj som saknades i en del program var att knappar till funktioner som användes frekvent saknades. Användaren blev tvungen att använda menyerna för att komma åt dessa funktioner vilket kändes oförklarligt.

Ett antal standarder hittades hos dagens modell- och testverktyg. Genomgående använde de sig av en progressbar för att illustrera testgenereringens förlopp. Detta fungerade väldigt bra eftersom användaren kontinuerligt fick återkoppling över hur stor del av testfallen som genererats samt kunde skaffa sig en uppfattning om hur lång tid det återstod innan alla testfallen skapats. När ett tillstånd markerades i modellprogrammen visades ett antal funktioner. En av dessa funktioner var att snabbt länka nya tillstånd till det markerade tillståndet. För att ändra texten i ett tillstånd eller en tillståndsändring skulle användaren markera eller dubbelklicka på det objekt som texten skulle ändras på. Funktioner för att rita modeller samlades tillsammans medan övriga funktioner sorterades på ett andra ställen. Bra att inte alla funktioner samlades tillsammans eftersom det hade blivit svårt att hitta en specifik funktion. För att markera vilka funktioner som inte gick att använda för tillfället användes grå färg. Ifall en knapp eller ett menyalternativ hade fått grå färg istället för originalfärgen indikerade det för användaren att det inte längre gick att använda.

Det skulle varit intressant att testa användargränssnittet i större utsträckning än vad det gjordes i det här examensarbetet. Antalet försökspersoner som testade prototyperna kunde varit större om mer tid hade funnits. Med den tid som gavs så satsades det på kvalitativa tester istället för kvantitativa. Det hade kanske varit utav intresse att testa om kulturella skillnader mellan olika världsdelar hade haft någon inverkan på resultaten från testerna. I mina tester var det endast svenskar som testade prototyperna. Något som jag inte lyckades med var att komma på ett helt nytt revolutionerade sätt att rita modeller på. Det skulle varit intressant att undersöka om det fanns något annat sätt att göra detta på mer än de traditionella metoderna.

4 Referenser

[1] Lunds tekniska högskola. (2009). Lunds tekniska högskola: LTH. Hämtat 11 juli från, Lunds tekniska högskola:
http://www.lth.se/

[2] System Verification. (2009). System Verification. Hämtat 11 juli 2009 från System Verification:
http://www.systemverification.com/sv_swe/

[3] Burnstein, Ilene. (2003). Practical Software Testing. New York: Springer.

[4] Object Management Group. (2009). Object Management Group - UML. Hämtat 11 juli 2009 från Object Management Group:
http://www.uml.org/

[5] Stevens, Perdita & Pooley, Rob. (2000). Using UML Software Engineering with Objects and Components. London: Addison-Wesley.

[6] Lauesen, Soren. (2002). Software Requirements - Styles and Techniques. London: Addison-Wesley.

[7] Gulliksen, Jan & Göransson, Bengt. (2002). Användarcentrerad systemdesign. Lund: Studentlitteratur.

[8] Norman, Donald A. (1988). The Design of Everyday Things. New York: Basic Books.

[9] Conformiq. (2009). Automated Test Design | Model-Based Testing (Conformiq). Hämtat 11 juli 2009 från Conformiq:
http://www.conformiq.com/

[10] AtYourSideConsulting. (2007). AtYourSideConsulting – Your provider for Software Quality Assurance Solutions. Hämtat 11 juli 2009 från AtYourSideConsulting:
http://www.atyoursideconsulting.com/

[11] Visual Paradigm. (2009). Increase productivity and enhance communication and collaboration efficiency by using UML. Hämtat 11 juli 2009 från Visual Paradigm:
http://www.visual-paradigm.com/

[12] Microsoft. (2009). Visio – startsida - Microsoft Office Online. Hämtat 11 juli 2009 från Microsoft:
http://office.microsoft.com/sv-se/visio/FX100487861053.aspx

[13] Microsoft. (2009). Windows XP - Microsoft Paint overview. Hämtat 11 juli 2009 från Microsoft:
http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/mspaint_overview.mspx?mfr=true

[14] Microsoft. (2009). Word – startsida - Microsoft Office Online. Hämtat 11 juli 2009 från Microsoft:
http://office.microsoft.com/sv-se/word/FX100487981053.aspx

[15] Microsoft. (2009). PowerPoint – startsida - Microsoft Office Online. Hämtat 11 juli 2009 från Microsoft:
http://office.microsoft.com/sv-se/powerpoint/FX100487761053.aspx

[16] Adobe. (2009). bildredigeringsprogram | Adobe Photoshop CS4. Hämtat 11 juli 2009 från Adobe:
http://www.adobe.com/se/products/photoshop/photoshop/

[17] Sun. (2009). Developer Resources for Java Technology. Hämtat 11 juli 2009 från Sun:
http://java.sun.com/

[18] Ruby. (2009). Ruby Programming Language. Hämtat 11 juli 2009 från Ruby:
http://www.ruby-lang.org/en/

[19] Eclipse Foundation. (2009). Eclipse.org home. Hämtat 11 juli 2009 från Eclipse Foundation:
http://www.eclipse.org/

[20] NetBeans. (2009). Welcome to NetBeans. Hämtat 11 juli 2009 från NetBeans:
http://www.netbeans.org/

[21] Wells, Don. (2006). Extreme Programming: A Gentle Introduction. Hämtat 11 juli 2009 från Wells, Don:
http://www.extremeprogramming.org/

[22] Sommerville, Ian. (2001). Software Engineering (6th ed.). London: Addison-Wesley.

[23] Wells, Don. (2008). User Stories. Hämtat 11 juli 2009 från Wells, Don:
http://www.extremeprogramming.org/rules/userstories.html

5 Bilagor

Bilaga A

Introduktion inför utvärdering av prototyper för SMART

Välkommen! Denna laboration ingår som en del i mitt examensarbete. Examensarbetet går ut på att utvärdera och designa ett användargränssnitt till ett modellbaserat testverktyg. Ett sådant modellbaserat testverktyg är SMART som för tillfället kodas hos ert företag System Verification. Detta examensarbete kommer sedan att ligga som grund för användargränssnittet till nyss nämnda program.

Du kommer att få testa två olika prototyper av användargränssnittet. Meningen med att testa två olika prototyper är att undersöka vilka strukturer som är att föredra i ett modellbaserat testverktyg. Innan själva testandet börjar kommer du att få en uppgiftsblankett där det står vad du ska göra. Under tiden som du testar dessa gränssnitt kommer du att filmas. Det filmade materialet kommer att vara till stor hjälp när jag ska analysera eftersom det är svårt att mäta testdata i realtid. Vidare kommer materialet endast att användas för personligt bruk så du behöver inte oroa dig över att någon annan ska få tillgång till materialet. Du uppmanas att i största möjliga mån att säga det du tänker högt eftersom det kommer att visa hur du tänkte när du gjorde ett visst val. Dessutom kommer det att vara till stor hjälp när jag analyserar materialet i efterhand. Efter att du testat prototyperna kommer du att få fylla i en enkät för att ge dina synpunkter på prototyperna. Det ges även tid för att muntligt redovisa för dina synpunkter som ej kom med i enkäten. Om du har några frågor angående denna information kan du ställa dem nu.

Bilaga B

Uppgiftsblankett för utvärdering av prototyper för SMART

Innan du får lov att börja testa prototyperna för SMART måste du läsa läsas igenom denna blankett. Eventuella frågor om blankettens innehåll bör ställas innan testet börjar då inga frågor kommer att besvaras under testets gång.

Instruktioner för testet:

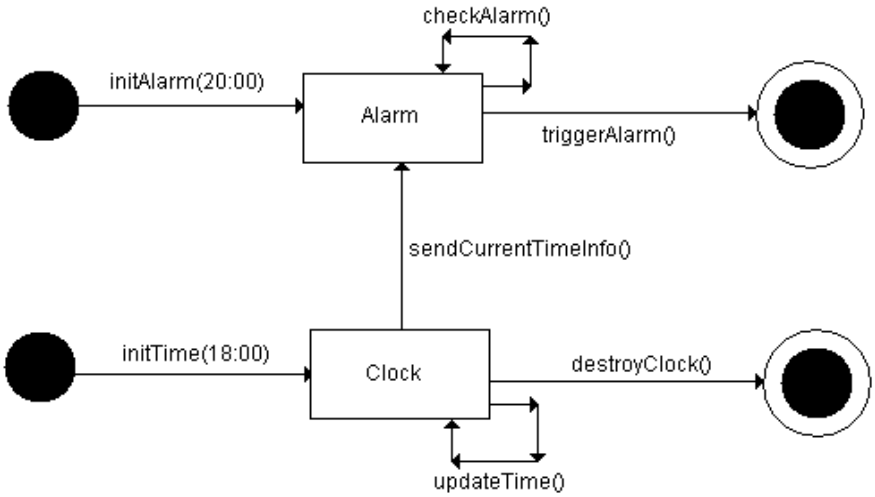
1. Läs igenom alla uppgifterna och kontrollera så att du förstår dem.
2. Tänk på att prototyperna har en del begränsningar i funktionalitet. Alla funktioner går inte att använda och en del fungerar inte riktigt som det är tänkt att de ska göra sen. De som kan komma att påverka din användarupplevelse är:

- När en ny modell skapas i prototyp Alpha kommer ytan för modellritande att nollställas. Du kommer också att förlora din föregående modell och den går tyvärr inte att få tillbaka.
- I prototyp Beta går det för tillfället inte att få en överblick över hur de olika modellerna hänger samman men detta kommer att åtgärdas till nästa version.
- När en modell sparas kommer den ej att sparas ”på riktigt” utan en endast en tom fil kommer att skapas. Det går ej att öppna en modell och få tillbaka det du sparade.
- Om du råkar göra fel när du ritar en modell går det tyvärr ej att ångra senaste utförda handling. Istället får du, om det är något allvarligt, börja om från början genom att skapa en ny modell eller radera den nuvarande.

3. Försök att tänka högt under testets gång så att dina tankegångar och funderingar tydligt framgår.
4. Uppgifterna ska utföras i den ordning som de står. Du kommer först att utföra alla uppgifterna på den ena prototypen och sen på den andra. Om du får börja med prototyp Alpha eller prototyp Beta avgörs via lottning som utförs av testledaren.
5. Skulle du råka fastna på en uppgift alldeles för länge, cirka 5 minuter, är det tillåtet att hoppa vidare till nästa uppgift.
6. Efter att prototyperna testats klart kommer du att få besvara en enkät och om det är något annat du vill ta upp angående prototyperna är det tillåtet att diskutera detta muntligt efter det att du fyllt i enkäten.

Uppgifter:

1. Rita en modell som ser ut enligt nedan. I prototyp Beta får du göra detta i Model_1.



Figur 1: Modell av en klocka som det går att sätta ett alarm på. Alarmet aktiveras när vald alarmtid inträffar. Bilden är tagen från en av prototyperna.

1. Spara modellen ni precis ritade som ”My_Model.mdl”.
2. Generera testfall utifrån modellen som heter ”My_Model_3.mdl”. Testfallen ska genereras för Java och övriga inställningar som ska göras är:

   - Algoritm ska sättas som ”Fastest Way”.
   - Täckning för testfallen ska sättas som ”State Coverage” och ”Method Coverage”.
3. Spara testfallen som genererades som ”My_Test_Suite.ts”.
4. Exekvera testfallen som finns i ”My_Test_Suite_3.ts”.
5. Stäng ner programmet.
6. Starta om programmet och bara testa diverse funktioner i fem minuter.
7. Stäng ner programmet och säg till testledaren att du är färdig.

Bilaga C

Svarsenkät för utvärdering av prototyper för SMART

Om någon fråga är svår att förstå, var inte rädd för att be testledaren om hjälp.

1. Antal års erfarenhet av testning?

❑ Mindre än 1 år ❑ 1-3 år ❑ 4-7 år ❑ 8-15 år ❑ Mer än 15 år

2. Har du tidigare erfarenhet av modellbaserad testning?

❑ Ja ❑ Nej

3. I vilken av prototyperna var det lättast att rita en modell?

❑ Prototyp Alpha ❑ Prototyp Beta

Varför? Motivera!

_____

_____

4. I vilken av prototyperna var det lättast att generera testfall?

❑ Prototyp Alpha ❑ Prototyp Beta

Varför? Motivera!

_____

_____

5. I vilken av prototyperna var det lättast att exekvera testfallen?

❑ Prototyp Alpha ❑ Prototyp Beta

Varför? Motivera!

_____

_____

6. Är knappar med text, användes i prototyp Alpha, eller knappar med ikoner, användes i prototyp Beta, bäst i ett modellbaserat testverktyg?

❑ Knappar med text, prototyp Alpha ❑ Knappar med ikoner, prototyp Beta
❑ En kombination av båda

Varför? Motivera!

---

7. Är ett uppifrån-och-ner-perspektiv, användes i prototyp Alpha, eller ett vänster-till-höger-perspektiv, användes i prototyp Beta, bäst för ett modellbaserat testverktyg?

❑ Uppifrån-och-ner-perspektiv, prototyp Alpha
❑ Vänster-till-höger-perspektiv, prototyp Beta

Varför? Motivera!

---

8. Upplevde du att du fick nog med återkoppling från prototyperna för att förstå vad det var som hände under användandet?

❑ Ja ❑ Nej

Varför? Motivera!

---

9. Upplevde du att prototyp Alpha var lätt att använda?

❑ Ja ❑ Nej

Varför? Motivera!

---

10. Upplevde du att prototyp Beta var lätt att använda?

❑ Ja ❑ Nej

Varför? Motivera!

---

11. Om du fick välja en av prototyperna, vilken skulle du då valt baserat på den totala användarupplevelsen?

❑ Prototyp Alpha ❑ Prototyp Beta

Varför? Motivera!

Bilaga D

Uppgiftsblankett för utvärdering av prototyp för SMART, iteration 2.

Innan du får lov att börja testa prototypen för SMART måste du läsa igenom denna blankett. Eventuella frågor om blankettens innehåll bör ställas innan testet börjar då inga frågor kommer att besvaras under testets gång.

Instruktioner för testet:

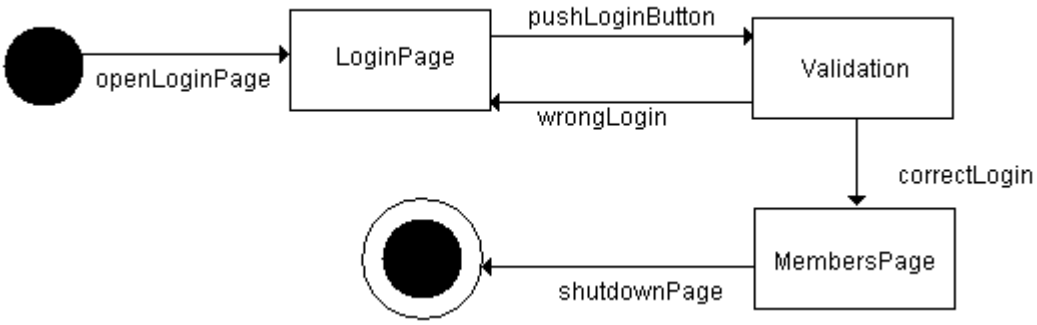
3. Läs igenom alla uppgifterna och kontrollera så att du förstår dem.
4. Tänk på att prototypen fortfarande har en del begränsningar i funktionalitet. Alla funktioner går inte att använda och en del fungerar inte riktigt som det är tänkt att de ska göra sen. De som kan komma att påverka din användarupplevelse är:

    - När en modell sparas kommer den ej att sparas ”på riktigt” utan en endast en tom fil kommer att skapas. Det går ej att öppna en modell och få tillbaka det du sparade.
    - Det går ej att flytta ett objekt som du skapat. Det finns ett sätt att komma runt detta och det är att använda sig av ångra-funktionen och sen sätta ut ett nytt objekt. Tyvärr måste man trycka en gång i modellytan efter att ha tryckt ”Ctrl+Z” för att ångra senaste. Samma gäller för ”Ctrl+Y”.

7. Försök att tänka högt under testets gång så att dina tankegångar och funderingar tydligt framgår.
8. Uppgifterna ska utföras i den ordning som de står.
9. Skulle du råka fastna på en uppgift alldeles för länge, cirka 5 minuter, är det tillåtet att hoppa vidare till nästa uppgift.
10. Efter att prototyperna testats klart kommer du att få besvara en enkät och om det är något annat du vill ta upp angående prototyperna är det tillåtet att diskutera detta muntligt efter det att du fyllt i enkäten.

Uppgifter:

2. Rita en modell som ser ut enligt nedan:



Figur 1: Modell av en hemsida som det går att logga in på för att komma till en särskild medlemssida. Bilden är tagen från prototypen.

3. Spara modellen du precis ritade som ”My_Model.mdl”.
4. Generera testfall utifrån en färdig modell som heter ”My_Finished_Model.mdl”. Tre stycken inställningar ska göras innan testfallen ska genereras. Testfallen ska genereras för Java och övriga inställningar som ska göras är:

   - Target ska sättas som ”Java”.
   - Algoritm ska sättas som ”Fastest Way”.
   - Täckning för testfallen ska sättas som ”State Coverage” och ”Method Coverage”.

5. Spara testfallen som genererades som ”My_Test_Suite.ts”.
6. Exekvera testfallen som finns i en färdig testsvit som heter ”My_Finished_Test_Suite.ts”. Följande testfall ska sättas så att de inte exekveras:

   - Testfall 6
   - Testfall 11
   - Testfall 18

7. Stäng ner programmet.
8. Starta om programmet och testa diverse funktioner i fem minuter.
9. Stäng ner programmet och säg att du är färdig.

Bilaga E

User stories för den andra implementeringsfasen. Följande *user stories* implementerades:

- Knapparna för verktygen nedan ändrades så att de fick ikoner istället för text:

    1. Ritverktygen, fick ikoner som visade det objekt som skulle skapas.

    2. Spara, fick standardikonen som liknar en diskett.

    3. Öppna, fick standardikonen som liknar en mapp.

    4. Skapa ny, fick standardikonen som liknar ett pappersark.

    5. Ta bort, fick en ikon som kombinerade standardikonen för att skapa en ny modell med ett rött kryss som skulle indikera att det hade motsatt verkan, dvs. att radera något.

    6. Starta, pausa och stoppa generering och exekvering av testfall, fick ikoner som liknade motsvarande knappar på en DVD-spelare.

- Texten som visade vilket verktyg som var valt för tillfället togs bort eftersom det kändes överflödigt. Den gröna bakgrundsfärgen på knappen till det valda verktyget markerade klart och tydligt vilket verktyg som var aktivt.

- Texten till knappen för att byta modell byttes från ”Change Model” till ”Set Model”. Däremot ändrades texten till ”Change Model” om en modell sattes. Likaså ändrades texten på knappen för att välja testsvit från ”Set Test Suite” till ”Change Test Suite” om en testsvit sattes.

- Knapparna för att starta och pausa generering samt exekvering av testfall kombinerades till en knapp. Om någon klickade på startknappen ändrades den till en pausknapp och om någon klickade på paus- eller stoppknappen ändrades den tillbaka till startknappen.

- Fixade ett fel som gjorde att informationen ifrån testgenereringen skrevs ut i informationsfönstret för testexekveringen.

- Fixade ett fel som gjorde att när en testsvit skulle laddas så sparades den istället. Fildialogen var en spara-dialog men ändrades till en ladda-dialog.

- Modeller, testsviter och testrapporter sparas med rätt ändelse även om ändelsen inte skrivs ut. Samtidigt sparas de inte som t.ex. My_Model.mdl.mdl om användaren väljer att lägga till ändelsen själv.

- När en testsvit öppnas visas det vilka testfall som ingår i testsviten. Det går dessutom att kolla på individuella testfall och se vad de testar samt välja om de ska ingå i testexekveringen.

- En ”OK”- och en ”Cancel”-knapp skapades i fönsterna där det gick att sätta algoritm och täckning. ”OK”-knappen sparar valen som gjorts och ”Cancel”-knappen återställer de val som var registrerade när fönstret öppnades.

- Textytan där algoritm- och täckningsval visades ändrades till enkla textrader så att ingen skulle tro att det gick att ändra dem via textytan.

- Fixade ett fel som gjorde att det gick att generera testfall även om ingen täckning valts.

- Knappen och menyalternativet för att visa statistik ifrån senaste testgenereringen togs bort. Kändes som att den informationen var onödig då all information som visades redan skrevs ut i informationsfönstret för testgenerering.

- Knappen för att visa testexekveringsinformation och knappen för att visa enbart fel ifrån senaste testexekvering kombinerades till en enda knapp. Detta gjordes eftersom endast den ena eller den andra informationen kunde visas på en och samma gång.

- Fixade ett fel som gjorde det omöjligt att ändra testsvit efter att en testsvit exekverats.

- Gjorde det möjligt att rita i flera modeller samtidigt. Dessutom blev det möjligt att ha flera modeller öppna på en och samma gång.

- Programmet frågar nu om man verkligen vill ta bort den modell som är markerad istället för att bara radera den utan någon fråga.

- Istället för att låta användaren hålla inne musknappen och placera ut ett objekt genom att släppa musknappen när det nått önskad position ändrades detta till att istället alltid visa en förhandsvisning av var objektet skulle hamna. Alltså visades objektet hela tiden och det var möjligt att flytta runt det. Det ”fastnade” inte på modellytan förrän användaren klickade på musknappen.

- Funktionen att spara ett helt projekt (modell, testsvit och testrapport) togs bort.

- En överblick av modellen med vilka tillstånd och tillståndsändringar som täcktes av de inställningarna som gjorts av användaren lades till i testgenereringsvyn.

- Knappar för att få information om de olika täckningsalternativen lades till i täckningsfönstret. Om en användare satte musen över dessa knappar visades ett tooltip som beskrev vad just det valet innebar. Dessutom lades det till knappar som öppnade fönster med extra inställningar, en knapp för varje täckningsalternativ. Knappen aktiverades först efter att ett täckningsalternativ blivit valt. Valet att lägga inställningar i vars ett eget fönster var för att slippa 50 olika inställningar i ett och samma fönster.

- Funktionalitet för att ångra samt att ångra det som ångrats lades till.

- Knappkombinationer för att byta modellverktyg lades till. Kombinationerna var enligt nedan:

    1. Tillståndsverktyget ”Shift+S” (state).

    2. Tillståndsändringsverktyget ”Shift+T” (transition).

    3. Startverktyget ”Shift+B” (beginning eftersom ”S” som i start redan var taget).

4. Slutverktyget ”Shift+E” (end).

5. Textverktyget ”Shift+T” (text).

User stories som ej hann fixas i tid men som ska åtgärdas:

- En överblick av modellen och vilket eller vilka tillstånd och tillståndsändringar som täcktes av ett specifikt testfall fattades i testexekveringsvyn.

- Funktionalitet för att klicka på informationsknapparna för täckningsalternativen så att ett tooltip visades lyckades ej implementeras.

- Fortfarande inget stöd för att markera ett objekt och därför ej möjligt att flytta ett objekt heller.

- Att kopiera, klippa ut eller klistra in ett eller flera objekt implementerades aldrig. Samma sak för att radera ett objekt.

- Ett verktyg för att markera flera objekt lades aldrig till.

- Tillståndsändringarna låstes ej till objekten. Det krävdes för mycket programmeringstid för att få detta att fungera på ett önskat sätt. För varje start- och ändpunkt av en tillståndsändring behövdes ett otal kontroller göras för att verifiera om tillståndsändringen var nära ett objekt eller inte.

- Textverktyget ändrades inte på grund av tidsbrist. Det gjordes däremot försök att lösa problemet men det skulle antagligen innebära att hela modellytan skulle behöva kodas om.

Bilaga F

User stories för den tredje implementeringsfasen. Följande *user stories* implementerades:

- Knappar för att markera eller avmarkera alla testfall lades till.
- Istället för att sätta ut text i efterhand skrivs det ut text direkt efter att ett tillstånd eller en tillståndsförändring skapats. Runt texten finns en grön rektangel och om användaren klickar innanför dess ram ska det vara möjligt att ändra texten. Att användaren var tvungen att klicka innanför ett markerat område skulle tydliggöra för användaren var hon skulle klicka för att ändra texten samt var hon skulle klicka för att markera objektet. Klickades det utanför textrutan skulle istället objektet markerats. Dock går det inte att ändra texten i prototypen p.g.a. tekniska begränsningar. Textverktyget som användes vid tidigare testningar skrotades med andra ord.
- Knappkombinationerna för de olika modellverktygen visas inom hakparantes i tooltipet för respektive verktyg.
- Muspekarna blev ändrade för modellverktygen. När ett start-, slut- eller tillståndsobjekt ska ritas ut visas ingen pekare alls och när en tillståndsändring ska ritas visas ett kors där mittpunkten är tillståndsändringens start- respektive slutpunkt.
- Lagt till ett verktyg för att markera flera objekt.
- Vid varje testgenerering och exekvering skrivs datum och tid för start ut samt vilken modell eller testsvit används.
- Ikoner för samtliga knappar har blivit uppdaterade och finputsade så att de inte såg så lika pixliga och fula ut.
- En överblick av modellen har lagts till i exekveringsvyn. Varje testfall kommer att visa vilka tillstånd och tillståndsändringar i modellen som testas.
- Informationsloggen i testexekveringsvyn har blivit uppdelad i en informationslogg och en resultatlogg.
- En knapp för att slumpmässigt ställa in täckningen av en modell har blivit tillagd i testgenereringsvyn.
- När programmet startas är tillståndsverktyget valt. Skapas en ny modellvy kommer det senast valda verktyget att sättas som aktivt verktyg i den nya vyn.
- Funktionerna för att importera eller exportera en modell har lagts till i filmenyn, dock är de inaktiva så de går ej att använda. Samma funktioner finns även i modellmenyn och även där ändrades de till att vara inaktiva. Detta eftersom dessa funktioner fanns med i kravlistan men glömdes bort i föregående prototyper. Att de gjordes inaktiva var för att de inte gick att använda.
- Menyalternativen för att välja modell, plattform, algoritm samt täckning modell togs bort från testgenereringsmenyn. De togs bort eftersom de aldrig användes utan istället gjorde användaren alltid dessa inställningar i testgenereringsfliken.
- Menyalternativet för att välja testsvit från testexekveringsmenyn togs likaså bort eftersom den inte heller användes.

- Stöd för att skriva ut en testrapport direkt från programmet lades till igen och återfinnes i filmenyn och testexekveringsmenyn. Denna funktion lades till eftersom den fanns i kravlistan men hade tagits bort i föregående prototyp.
- Infoknapparna som användes för att visa ett tooltip, där en täckningsbeskrivning gavs, skrotades och istället sattes tooltipen ut på kryssrutorna.
- En prototypinställning för att visa hur ett objekt skulle se ut när det markerats lades till. Om ett objekt markerades skulle det finnas gröna rektanglar i hörnen av objektet som skulle fungera som snabbgenvägar. Användes någon av dessa snabbgenvägar skulle det ritas ut ett tillstånd med en tillståndsändring från det markerade objektet i den gröna rektangelns riktning.
- Diverse buggar av mindre betydelse åtgärdades.
- Diverse finjusteringar för att snygga till gränssnittet genomfördes.

User stories som ej hann fixas i tid eller som ej gick att genomföra p.g.a. tekniska begränsningar återfinnes nedan:

- Det gick inte att skapa en tillståndsändring genom två musklick men skulle implementerats om mer tid hade funnits.
- Programmet frågade ej om osparat material skulle sparas innan det stängdes ner.
- Ett testfall valdes eller valdes bort om användaren klickade på texten istället för kryssrutan. Självfallet skulle ett testfall endast väljas eller väljas bort om det klickades på kryssrutan. Detta var en teknisk begränsning hos kryssrutorna som användes för detta.
- Ej möjligt att markera ett objekt genom att klicka på det.
- Inget stöd för att radera, kopiera, klistra in eller klippa ut objekt.
- Ej möjligt att välja mer än en algoritm vid testgenerering.
- Tip-of-they-day vid allra första starten av programmet implementerades aldrig på grund av tidsbrist. Samma sak gällde för hjälpmenyn.
- Stöd för att grafiskt visa vilka testfall som resulterade i rätt eller fel samt vilka testfall som hoppades över. En idé var att låta godkända testfall symboliseras av en rund grön cirkel, icke godkända testfall av ett rött kryss och testfall som ej exekverades skulle representeras av ett blått horisontellt streck.
- Miniatyrer av modellerna skulle visats i trädstrukturen så att användaren lätt kunde avgöra vilken modell hon ville använda.