

Sudoku til undervisningsbrug

Emil Erik Hansen Julian Møller Klaes Bo Rasmussen
Steen Nordsmark Pedersen

11. juni 2007

Indhold

1	Formalia	4
2	Formål	4
3	Indledning	4
4	Problemformulering og afgrænsning	5
4.1	Baggrund	5
5	Analyse	6
5.1	Målgruppe	6
5.2	Analyse af problem	6
5.2.1	Kategorien Sudoku	7
5.2.2	Kategorien Brugbarhed	7
5.2.3	Kategorien Motivation	8
5.2.4	Kategorien Teknik	8
5.3	Begrænsninger	9
6	Design af løsning	10
6.1	Overordnet design	10
6.2	Model-View-Control (MVC)	11
6.2.1	Model	11
6.2.2	View	12
6.2.3	Control	12
6.3	Observer og Observable	12
6.4	Datarepræsentation	13
6.5	Algoritmer	13
7	Grafisk design	14
7.1	Vælg Sudoku	14
7.2	Sudoku	14
7.3	Tillykke	15
7.4	Maskot	16
8	Teknisk beskrivelse	16
8.1	Programlogik	16
8.2	Logik i Control	17
8.3	Sudokuløser	17
8.3.1	solveField	19
8.3.2	solverLevelOne	19
8.3.3	solverLevelTwo	19
8.4	Sudokumatematik	20
8.5	Hjælpefunktionen	20

8.6	Generator	21
9	Funktionstest	21
9.1	Unittest	22
9.2	Blackbox	22
10	Brugertest	23
10.1	Tilgængelighed, tekst, grænseflade og motivation	23
10.2	Linuxkompabilitet	26
10.3	Appletkompatibilitet	27
11	Konklusioner	27
12	Bilag	29
12.1	Testspecifikation	29
12.2	Designdokument	29

1 Formalia

Projekttitel	Sudoku til undervisningsbrug
Gruppenummer	4
Studerende	Emil Erik Hansen Julian Møller Klaes Bo Rasmussen Steen Nordsmark Pedersen
Instruktor	Dennis Franck
Underviser	Georg Strøm

2 Formål

Den følgende rapport er en besvarelse på den stillede opgave til kurset Førsteårsprojekt på Datalogisk Institut, Københavns Universitet, år 2007, under vejledning af Dennis Franck. Opgaven gik ud på at konstruere et program der kan stille løsbare Sudoku-opgaver til børn fra 1. til 3. klasse i folkeskolen. Programmet skal være let tilgængeligt samt underholdende og sjovt for børnene. Der skal derudover være mulighed for at børnene undervejs i spillet skal kunne få hjælp, såfremt de sidder fast, og ikke kan komme videre. Derudover skulle det være muligt at vælge forskellige sværhedsgrader, for at kunne ramme en større intellektuel målgruppe af børnene.

3 Indledning

Der var mange grunde til at vi ønskede netop dette emne.

Der var mulighed for at prøve forskellige aspekter af software-udvikling. Der skulle laves en pæn, underholdende og overskuelig brugergrænseflade. Der skulle arbejdes med matematikken og algoritmerne bag sudoku og begge ting skulle tilpasses således at det tiltalte og var brugbart for børn. Derudover syntes vi at Sudoku er en god måde at indlære logisk tænkning og overblik, noget som er en god egenskab senere i de matematiske fag. Sudoku var desværre ikke fremme dengang vi selv var på de klassetrin, og vi syntes det virker som passende at have det sjovt mens man indlærer sådanne færdigheder.

Forudsætningerne for at læse denne rapport er kendskab til programmeringssproget Java, samt simpel sudoku teori som kan findes på <http://www.sudokumester.dk/?ID=guide>.

Sudoku bliver brugt i folkeskolens matematikundervisning. Børnene får udleveret sudokuopgaver på papir, som de skal løse, enten i klassen eller som hjemmearbejde. Denne metode har visse ulemper. Børnene kan ikke få hjælp hvis de sidder fast, hvis ikke der er en anden person der kan løse sudokuen i nærheden. Desuden kan de komme ud for at have løst en sudoku forkert, hvis

de ikke har været opmærksomme på eventuelle fejl. Et program til computeren eller internettet, der kan generere og vise sudokuer, samt hjælpe børnene hvis de har brug for det, kan derfor være en hjælp for både børn og lærere. Flere sværhedsgrader giver også sudokuerne en bredere målgruppe, da børnene kan få genereret en der svarer til deres faglige niveau. Endvidere kan børnene nemt få flere sudokuer, hvis de har lyst, så de ikke er begrænset til antallet de har fået udleveret i skolen. Den indbyggede hjælpefunktion medvirker også til at børnene forhåbentlig kan løse sudokuerne uden assistance fra voksne, så børnene ikke er afhængige af f.eks. forældres tid til at lave lektier med dem.

4 Problemformulering og afgrænsning

4.1 Baggrund

En sudoku er en spilleplade hvor den mest normale størrelse 9 gange 9 felter¹. Disse felter er, når spillet er færdigt, udfyldt med tallene fra 1 til 9. Pladen består af rækker, kolonner og kvadranter. Hver af disse er på 9 felter, og der er 9 af hver af dem på en fuld spilleplade. Rækker og kolonner er ikke så svære at skelne, men kvadranterne er lidt anderledes. Disse består af små spilleplader på 3 gange 3 felter, indbygget i den store spilleplade som så består af 3 gange 3 kvadranter. Et eksempel på en tilfældig sudokuplade kan ses på Figur 1.

2	5			3		9		1
	1				4			
4		7				2		8
		5	2					
				9	8	1		
	4				3			
			3	6			7	2
	7							3
9		3				6		4

Figur 1: En sudoku spilleplade

I hver række, kolonne og kvadrant skal der være præcis ét af hver af de 9 tal². Med denne viden skal man kunne udfylde en sudoku hvor ét eller flere af felterne er blanke. Jo færre tal der er på pladen, jo sværere bliver sudokuen at løse og man skal til tider tænke fremad for at kunne finde ud af hvordan man skal løse den. Det specielle ved at generere sådan en sudoku er at det

¹Sudokuer kan genereres i mange forskellige størrelser. Den mest normale er dog 9 gange 9 felter, også kaldet 3 gange 3 (kvadranter)

²Disse "regler" er i resten af rapporten omtalt som "sudokureglerne".

ikke er en rigtig sudoku hvis man kommer til at fjerne et tal fra pladen der gør at sudokuen har flere end én mulig løsning. Hvis man begynder med en tilfældigt genereret fuld plade og derefter fjerner tal fra den er man således nødt til at være sikker på at man ikke fjerner tal der åbner for flere løsninger.

En spiller skal således kunne få en sudoku plade serveret med en justerbar sværhedsgrad og have mulighed for at udfylde felterne på pladen samt ændre deres egne placerede tal eller fjernelse af samme. Brugeren skal have mulighed for at få hjælp til løsningen, eller til at komme videre hvis vedkommende sidder fast. Dette skyldes det basale krav om at det skal være muligt at få hjælp uden at skulle spørge en udefrakommende. Man skal ligeledes kunne få af vide når spillet er færdigt, få et nyt spil på et hvert tidspunkt og afslutte programmet når man har lyst. Programmet skal virke indbydende og sjovt samt let tilgængeligt for mindst den givne brugergruppe.

5 Analyse

5.1 Målgruppe

Målgruppen for vores projekt er 1. - 3. klasse elever. Da målgruppen er 1. - 3. klasse blev der også udformet krav ud fra det standpunkt. Der er blandt andet krav om brugbarhed, der skal gøre det muligt kun at betjene programmet udelukkende ved hjælp af musen.

5.2 Analyse af problem

Fra interessenten fik vi stillet kravene om at skal være muligt at:

- ... generere løsbare Sudokuer.
- ... vælge forskellige sværhedsgrader.
- ... få hjælp til at komme videre når brugeren er kørt fast.
- ... programmet skal være let anvendeligt.
- ... der ikke kræves for høje læsefærdigheder.
- ... programmet skal virke underholdende og sjovt.

Ovenstående krav fra interessenten, blev delt op i katorierne *sudoku*, *brugbarhed*, *motivation* og *teknik*. Vi har tilføjet yderligere krav, da vi syntes de var nødvendige for at opnå et tilfredsstillende produkt. Det skal give give børnene det faglige indhold i programmet, samtidigt med at de bliver underholdt. Derudover er de tekniske krav for selve computeren sat lavt således at det langt fra er nødvendigt at skolerne har nye og hurtige computere eller komplet opdateret software.

5.2.1 Kategorien Sudoku

Programmet skal naturligvis kunne generere sudokuer, således at vi sikrer os, at der altid er nye udfordringer til brugeren. At have en generator fremfor en enorm mængde predefinerede sudokuer liggende sparer ikke blot en masse plads i koden, men forhindre at man løber tør for nye sudokuer. En dybere teknisk beskrivelse af generatoren kan læses på side 21.

I en sudoku er der forskellige måder at gøre niveauet mere udfordrende. Eksempelvis hvor mange felter der er udfyldt i forvejen, størrelsen af sudokuen samt hvilken sudoku metode der er brugt til at fjerne tallene.³ Vi valgte at sætte sværhedsgraderne som et underkrav til at der skulle kunne genereres sudoku, da det ikke ville give mening at definere sværhedsgrader på noget, der i forvejen var umuligt at løse. Vi valgte at regulere sværhedsgraden ved at ændre på antallet af felter der bliver fjernet.

Børnene skulle derudover være i stand til at få hjælp til at løse deres sudoku. Denne funktion er utroligt åben, da typen af hjælp der kan gives er meget forskellig. Vores eksempel fra kravspecifikationen med at markere felter der kunne løses udfra informationen der kunne ses, besluttet vi os for at følge. Grunden til dette var at vi ikke ville have at brugeren skulle kunne løse sudokuen udelukkende ved brug af hjælpefunktionen. En yderligere idé vi senere fik var at, i tilfælde af fejl, skulle de fejl udfyldt felter markeres med rød farve når hjælpefunktionen blev kaldt. Hjælpefunktionen er videre beskrevet på side 20.

5.2.2 Kategorien Brugbarhed

Vi satte det som det primære krav at programmet skulle være let anvendeligt for brugerne, dvs. at der såvidt muligt ikke måtte være tvivl om hvordan programmet videre forløber, og hvordan de forskellige handlinger udføres. Dette krav er umiddelbart meget tæt opad kravet stillet af interessenten, dog med den forskel at vi har begrænset vores betydeligt ved at sætte en begrænsning på, der kræver at brugeren højst må bruge 5 minutter førend de skal kunne betjene programmet. Grunden til at vi satte denne begrænsning, var at vi synes begrebet "let anvendeligt" var meget løst, og krævede en klarere definition. På baggrund af det er dette krav et superkrav til de to følgende krav i denne kategori.

Naturligvis optræder der noget tekst i programmet, og denne vil vi også gerne have skal opretholde brugbarhedskravet ved at være let forståelig og ikke kræve de store sprogfærdigheder af brugerene. For at have et nogenlunde udgangspunkt, sigtede vi efter at LIX-tallet maksimalt skulle ligge på 8, men det ville være godtaget hvis det viste sig at børnene forstod den skrevne tekst. Dette fik vi testet ved vores brugertest, hvor børnene blev overvåget mens de brugte programmet, og udspurgt om de forstod teksten. Disse er beskrevet

³Læs mere om metoderne på side 17.

yderligere i afsnittet "Tilgængelighed, tekst, grænseflade og motivation" på side 23.

Styringen af programmet valgte vi selv at tilføje som et selvstændigt krav, da vi syntes at det ville hjælpe betydeligt på hvor let anvendeligt programmet var, hvis brugeren kunne nøjes med udelukkende at bruge musen. Alternativet ville være at tastaturet skulle i brug, og situationen hvor brugeren var nødt til at fjerne fokus fra skærmen kunne opstå. En af de mål ved let anvendelighed vi ønskede at opnå med dette var at brugeren med det samme skulle kunne blive fanget. I almindeligt sudoku på papir er der til sammenligning ikke brug for at personen der løser den kigger andre steder hen end på spillepladen.

Vi valgte ligeledes at tilføje et selvstændigt krav for at det skulle være muligt at slette de tal man indsætter på spillepladen. Dette kan være relevant i tilfælde af at der er fundet fejl på et eller flere felter, evt. ved brug af hjælpefunktionen, eller hvis brugeren har ombestemt sig i sit valg af talplacering. Ligesom man i en sudoku på papir kan viske sine tal ud, hvis man har lavet en fejl. Dette krav er dog lidt af en selvfølgelighed, da brugerne naturligvis vil lave fejl på et eller andet tidspunkt. Det kunne eksempelvis ske at et tal fejlagtigt blev sat ind på grund af et fejltryk.

5.2.3 Kategorien Motivation

Det første af de to krav vi besluttede os for var at brugeren skulle have mulighed for at se hvordan vedkommende havde klaret sig efter hvert spil. Dette inkluderer hvor langt tid der er brugt på at løse sudokuen, dens sværhedsgrad, antal fejl der er brugt og hvor meget hjælp programmet har givet. Dette fandt vi ville være en god ide, da konkurrence er et incitationselement der får børnene til at anstrenge sig for at løse opgaven bedre end før, idet de både kan sammenlignes med tidligere resultater fra sig selv og andre brugere.

Den anden del af kategorien inkluderer vores maskot, som skulle virke sød og venlig overfor børnene. Den skulle derudover integreres med hjælpefunktionen, således at maskotten blev brugt til at formidle beskeder fra programmet til brugeren. Dette er en god måde at formidle informationen på.⁴

5.2.4 Kategorien Teknik

Vi valgte at basere det tekniske grundlag på en platform uafhængig af styresystem, da det dermed ikke er nogen hindring at bruge f.eks. Linux - et problem man ellers ofte kan komme ud for i mange kommercielle softwareprodukter.

⁴Vi lærte senere at der var visse problemer med maskottens kommentarer, som kan læses i brugertest afsnittet på side 23.

Foruden det, valgte vi at gøre det til et krav at programmet ligeledes skal kunne køres i en internetbrowser, istedet for at programmet skal ligge på hver enkel computer der skal benytte programmet. Dette vil være en klar fordel når en hel skoleklasse skal bruge programmet. Da det kan ligge på en (intern) internetside, og som brugerne bare kan åbne. Dette vil også kunne øge programmets udbredelse mere end en eksekverbar fil vil kunne, idet det er nemmere at åbne en internetside i en browser i stedet for at skulle finde programmet, downloade det og derefter køre det.

De to ovenstående punkter fik os til at vælge at basere vores program på Java 1.5-plattformen, da programmet dermed både vil kunne afvikles som eksekverbar fil men også som applet i en internetbrowser. Grunden til valget af version 1.5 er at den er mere udbredt end den i skrivende stund nyeste 1.6-udgave, og vi ikke vil have at programmets krav gør det umuligt at afvikle programmet fordi brugerne ikke står med de nyeste udgaver af deres software, hvilke grunde der end måtte ligge bag.

5.3 Begrænsninger

Der er mange muligheder for at tilføje moduler til sudokuspillet.

Vi har blandt andet overvejet mulighederne for en topcoreliste. Det kunne ske ved at resultaterne gemmes i en fil på harddisken, eller at der blev oprettet en database på en server hvor man så uploadede resultatet til. Listen over resultater skulle primært inkluderes for at give videre incitament til at fortsætte med at spille spillet, for - naturligvis - at kunne præge topscorerlisten. Selvom denne funktion ville være både sjov, og god for spillet at have med, går den unødvendigt uden for opgavekravene og derved er det ikke noget vi vil bruge tid på at implimentere.

I forbindelse med ovenstående overvejede vi til slut i udviklingsprocessen også at lave en pointalgoritme, der omdannede spilstatistikkerne "Antal fejl", "Antal hjælp brugt", "Sværhedsgrad" samt "Brugt tid" til en samlet pointscore, der kunne bruges som sammenligning brugere imellem og samtidig danne basis for topscorerlisten.

Vi valgte også at begrænse vores spilleplade til "standardstørrelsen" på 9x9 felter, idét vi ikke mente at den ekstra tid det eventuelt ville tage at implementere andre størrelser i generatoren var tiden værd, når sværhedsgraden i forvejen kan gradueres så meget som den kan på en 9x9 spilleplade.

Programmets maskot valgte vi også at lave til et statisk objekt. I starten overvejede vi at lave det animerbart, så forskellige tegninger kunne vises alt efter brugerens aktivitet. Den smule der var at vinde, fandt vi dog ikke af en tilpas størrelse til at tidsforbruget var retfærdiggjort.

Hvis programmet en dag skal videreudvikles er ovenstående punkter dog klart på toppen af en imaginær prioritetsliste.

En mere teknisk begrænsning ligger i vores sudokuløser, der bruges til at hjælpe med at generere sudokuplader. Denne kunne være udvidet med funk-

tionalitet, så den kunne forsøge at iterere over et "gættetetræ", hvis det ikke ud fra udelukkelse og brug af sudokureglerne var muligt at placere et tal. Hvis en gren ville vise sig at være ugyldig skulle funktionen altså hoppe tilbage til forrige samling og forsøge en anden gren. Gættefunktionalitet i løseren ville give os mulighed for at fjerne flere tal fra en Sudokuplade og dermed kunne generere sværere Sudokuer. Dette blev dog fravalgt pga. målgruppens alder kombineret med det faktum at vores højeste sværhedsgrad er tilstrækkelig for størstedelen af eleverne i 1.-3. klasse.

6 Design af løsning

6.1 Overordnet design

Vi har valgt at lave vores program så det kan afvikles med Java 1.5 fra en eksekverbar JAR fil eller som applet i en browser (Internet Explorer version 5.5 eller derover samt Mozilla Firefox version 1.5 eller derover) på Linux og Windows. Dette er valgt da vores brugere ikke altid har den nyeste version af hverken Java eller de browsere de nu bruger. Specielt på skoler, som programmets målgruppe ofte vil befinde sig på under afviklingen.

Der er flere hovedpunkter designmæssigt. Det ene og et af de vigtigste er græsefladen. Da det er et program for børn skal den være simpel og let at bruge. Derudover skal den være tillokkende og flot for at holde børnenes interesse.

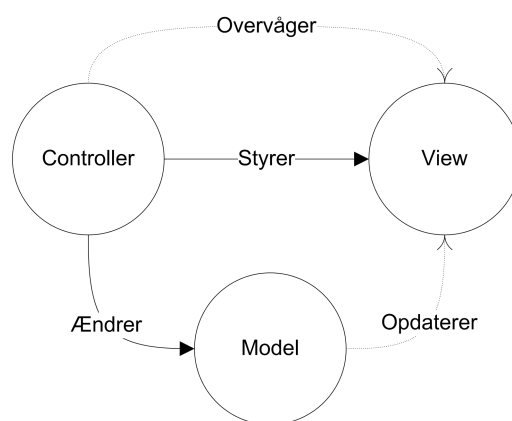
Det andet hovedpunkt er vores sudokugenerator. Den skal kunne generere korrekte løsbare sudokuer men samtidig skal det være tilfældigt genererede og det skal være muligt at tilføje flere sværhedsgrader. Desuden var det et mål at generatoren skulle være så hurtig, at man ikke skulle vente på at nye spil blev genereret, men blot kunne fortsætte direkte når en sværhedsgrad var valgt. Hvis generatoren havde været for langsom ville det formentlig have afskrækket brugere fra at bruge programmet, fordi mange er utålmodige når de sidder foran en computer - oplysningerne skal helst komme frem med det samme.

Endvidere skal systemet være let at udvide med ny funktionalitet, hvilket er en af grundene til at vi har valgt at skrive i et objektorienteret sprog. Vi har også gjort brug af Model-View-Control-designparadigmet, da det tillader nem udvidelse af én komponent uden at redigere nævneværdigt i andre. Derudover har vi så vidt muligt gjort brug af abstrakte klasser og **interfaces** i vores programdesign, for at det skulle være nemt f.eks. at tilføje indstillinger eller andre grænseflader end de to implementerede (Java applikation og Java applet). Det gør det nemlig muligt for kommende udviklere blot at skrive klasser der læner sig op af de i forvejen lavede **interfaces**, hvormed al allerede eksisterende funktionalitet meget gerne skulle fungere med det samme.

6.2 Model-View-Control (MVC)

Vores design er bygget op omkring den klassiske MVC-model.

Den indeholder de tre moduler Model, View og Control. Dette valg giver gode muligheder for videreudvikling af programmet da MVC modellen giver muligheder for at udskifte dele af programmet uden det påvirker de andre. Det er desuden nemt at vedligeholde, da al programlogikken ligger i én separat del, komplet adskilt fra programmets udseende og data. på den måde kan man nemt f.eks. skifte spillets “motor” uden at skulle ændre hverken programlogikken eller den grafiske grænseflade. Det fulde design dokument uddyber designet og kan findes som bilag på side 29. Figur 2 viser det overordnede samspil mellem de tre dele og viser hvorledes **Control** styrer hvad der skal vises i **View** samtidig med at komponenten ændrer dataen i **Model**.



Figur 2: Model-View-Controller samspillet

6.2.1 Model

Model indeholder selve “regnemaskinen” i programmet. Det er i dette modul at sudokuen bliver genereret. Det er også her vores sudokuløser huserer. Vi bruger sudokuløseren dels som hjælpefunktion til generatoren og dels som baggrund for hjælpefunktionen. **Model** indeholder oplysninger om de data der forekommer i programmet (f.eks. hvordan sudoku-spilleplader ser ud). Modulet indeholder også metoder til at ændre disse data. Endvidere indeholder **Model** også indstillinger for sværhedsgraderne samt en klasse der holder styr på spilstatistikker. Der er også basale matematikmetoder at finde her. Matematikmetoder der f.eks. omregner et felt-Id til et koordinat på en vilkårlig sudokuplade eller kan finde Id-numrene på alle tal i et felts kvadrant, kolonne eller række.

6.2.2 View

View er den grafiske repræsentation af programmet. Det indeholder hele den grafiske brugergrænseflade og sørger for at dette bliver vist korrekt. **View** henter oplysninger fra **Model** om hvilke data der skal være i de viste dele. **View** bliver kontrolleret af **Control**, så der skiftes mellem de forskellige skærbilleder. **View** består af tre primære skærbilleder, "Nyt spil", "Spillepladen" og "Tillykke". Skærbilleder der er beskrevet yderligere i afsnittene startende på side 14.

6.2.3 Control

Control er det modul der styrer programmet. Det er **Control** der bliver kaldt når programmet startes. Det sender besked til **Model** om at generere en løslar sudoku hvorefter det kalder **View** modulet til at lave en grafisk repræsentation som brugeren kan interagere med. Det er også i **Control** at handlinger, der skal udføres når man trykker på knapper, menupunkter o.l., ligger. Disse bliver vedhæftet **View** når den grafiske repræsentation oprettes. **Control** sørger også for at dataene i **Model** bliver ændret sammen med den grafiske repræsentation i **View** når brugeren foretager en handling i programmet.

6.3 Observer og Observable

Udover MVC paradigmet, har vi også implementeret Java-mønstrene **Observer** og **Observable**. Disse gør det muligt for klasser at overvåge andre klasser. På den måde kan man i en **Observable**-klasse gøre alle **Observer**-klasser der overvåger den pågældende klasse opmærksom på at indholdet i klassen er ændret. Derved kan **Observer**-klasserne selv sørge for at gøre hvad de nu har brug for med den nye data.

Vi har implementeret det så `model.Board` implementerer **Observable** mens `view.Board` implementerer **Observer** og dermed overvåger `model.Board`. På den måde bliver spillepladen gentegnet og fyldt med frisk data når der startes et nyt spil.

På grund af tidspres var det denne lille og forholdsvis simple metode vi gjorde brug af, og ikke en større udgave hvor f.eks. hvert felt på spillepladen implementerede **Observer** og overvågede det specielle stykke data i `model.Board` som de viste. Hvis dette var blevet implementeret kunne den del af programlogikken i **Control**, der står for at opdatere spillepladen når dataene ændrer sig være sparet væk, idét Javas interne teknikker kunne have taget sig af det. Der skulle dog naturligvis stadig være metoder i hhv. den observerende og observerbare del, der rent faktisk tog sig af at ændre tallene, men logikken kunne være flyttet ud til dér, hvor den rent faktisk bliver brugt.

6.4 Datarepræsentation

Vores sudokuplader er i programmet repræsenteret som et simpelt array af heltal, med en fastsat størrelse.

Vi har valgt denne primitive datastruktur da den simplificerer de matematiske funktioner, der er nødvendige for at kunne operere med sudokuplader, fordi hvert element i et array har en værdi og et Id-nummer. Desuden er et array en af de hurtigste datatyper i Java, hvis man ved på hvilken plads elementerne skal hentes fra eller indsættes på, og da dette er veldefineret i et sudokuspil fandt vi det sensibelt at gøre brug af arrays. Høj hastighed var også et vigtigt kriterie for os, i forbindelse med vores sudokugenerator, så på dette punkt var det også naturligt at bruge arrays af heltal.

Naturligvis kunne vi have oprettet et array af felt-objekter, men eneste umiddelbare fordel vi så ved dette var udvidelsen af **Observer-Observable**-designparadigmet, som beskrevet i afsnit 6.3.

6.5 Algoritmer

I vores indledende udviklingsfase undersøgte vi diverse muligheder for at bruge en i forvejen lavet sudokugenerator og -løser. Vi fandt dog ingen med passende licenser, der samtidig var lette at implementere i vores eget program.

Derfor endte vi med at tage beslutningen om selv at udvikle både generatoren og løseren. På denne måde var vi også sikre på at vi ville ende med singulære sudokuer. Derudover kunne vi også selv bestemme hvordan de forskellige sværhedsgrader skulle håndteres og gradueres, så vi nemt kunne ende med sværhedsgrader der passer til vores målgruppe.

Dette betød dog også at vi fik tilføjet en væsentlig arbejdsbyrde, idet vi også skulle konstruere hjælpefunktioner til at regne på Sudokuplader samt selv finde ud af hvordan sværhedsgrader kunne defineres.

Alt i alt føler vi dog at alt dette har givet os en mere solid løsning, fordi vi selv har kontrol over alle aspekter af programmets bagvedliggende funktionalitet.

Vi har valgt kun at kunne generere en størrelse sudoku plade. Vi har været nødt til selv at skrive algoritmerne til dette da vi ikke kunne finde nogen tilgængelige algoritmer med passende licens aftaler. Dette gør vi ved først at generere en tilfældig, fyldt sudokuplade. Det sker ved at vi blander en gyldig sudokuplade vilkårligt rundt. Man må dog kun bytte rækker med rækker i samme kvadranttrække og kolonner med kolonner i samme kvadrantkolonne⁵.

Der fjernes nu et tal af gangen. Når et tal fjernes beder vi vores sudoku løser om at løse det givne felt, hvis den kan lade vi feltet være tomt, derefter

⁵Da en sudoku er opdelt i kvadranter er kvadranttrækker og kvadrantkolonner defineret ved tre sammenhængende rækker eller kolonner der deler de samme tre kvadranter på spillepladen.

fortsætter til vi har en tilfredsstillende sudoku plade.

Denne fremgangsmåde kræver således en algoritme til ikke bare at kunne generere, men også løse sudokuer. Til gengæld kan den samme algoritme bruges til at hjælpe brugeren undervejs i spillet, da der altid vil være mindst ét felt denne løser kan løse.

7 Grafisk design

Det grafiske design er en væsentlig del af vores projekt. Da det skal appellere til børn er det meget vigtigt at brugerne kan lide det og føler det imødekommende. Dette kan dog kun afgøres i brugertest, dog er det muligt at gøre tingene simple, men flotte før det er muligt at teste det. Vores brugergrænseflade består af tre skærbilleder "Vælg Sudoku", "Sudoku" og "Tillykke". På alle skærbilleder optræder vores maskot, som vi har gjort meget for udgør en stor del af spiloplevelsen.

Alle farver er så vidt muligt præget af en pastelbaseret farvepalet, dels for at gøre knapper og overskrifter tydelige, dels for at de skal appellere til brugerne.

Knapper har vi valgt at holde i store, runde former, alle med en kort tekst. Det skal hurtigt gøre brugerne i stand til instinktivt at kunne afkode hvilke konsekvenser der vil være når de trykkes på en knap.

Overskrifter er holdt i runde, bløde skrifttyper med en forholdsvis stor skriftstørrelse. Teksten i overskrifterne er som al anden tekst i programmet holdt så kort og simpel som mulig.

På spillepladen har vi valgt at gøre forudplacerede tal grå, for tydeligt at adskille dem fra de tal brugeren placerer undervejs, der er sorte. Dette er valgt for at gøre det klart over for brugeren at der ikke forekommer nogen handling når man trykker på de i forvejen placerede tal, mens det er muligt at gøre noget ved at trykke på de tal man selv har placeret.

7.1 Vælg Sudoku

På dette skærbillede har brugeren valget mellem en "Let", "Mellem" og "Svær" sudoku. Når en af disse er valgt fortsættes der til næste skærbillede. Det er denne skærm der vises som det første når brugeren starter programmet.

Figur 3 viser hvordan dette skærbillede ser ud.

7.2 Sudoku

Sudoku skærbilledet består af en menulinje, en maskot med dertilhørende hjælpetekst en sudoku-spilleplade og to knapper. Fra menulinjen er det muligt at starte et nyt spil, lukke spillet, få hjælp til at løse sudokuen, og at få information om hvor de generelle sudoku regler kan findes.



Figur 3: Skærmen hvor brugeren kan vælge en sværhedsgrad

På sudoku-spillepladen kan man ved hjælp af musen indsætte og fjerne tal.

I højre side finder man de to knapper "Hjælp" og "Nyt spil" som giver brugeren mulighed for at få hjælp til at løse sudokuen samt at komme tilbage til det foregående skærbillede og derved starte en ny sudoku.

Hjælpe- og informationstekst forekommer i taleboblen ud for program-mets maskot, fåret Dolly. Det er her information om fejl, hjælp til løsbare felter og lignende forekommer.

Figur 4 viser hvordan dette skærbillede ser ud.



Figur 4: Sudoku spillepladen

7.3 Tillykke

Det sidste skærbillede er "Tillykke" billedet. Der vises statistik for den løste Sudoku samt en knap til at starte et nyt spil. Statistikken indeholder antal fejl, antal 'hjælp' brugt og tidsforbruget. Statistikteksten er holdt i simple udtryk, for at gøre den let forståelig.

Figur 5 viser hvordan dette skærbillede ser ud.



Figur 5: Tillykke-skærmen der vises når en Sudoku er løst

7.4 Maskot

Maskotten, fåret Dolly, følger brugeren hele vejen gennem spillet. Fåret er primært til for at gøre det sjovt for børnene mens de spiller, men det udøver også en hjælpefunktion idet det kommer med simple forklaringer til de ting der sker på skærmen. Ligesom resten af det grafiske i spillet, er fåret ligeledes tegnet i pastelfarver, og med et let tegnefilmspræg, for at passe ind i resten af programmets stil.

8 Teknisk beskrivelse

8.1 Programlogik

Når brugeren starter programmet aktiveres **Control**, der opretter en **JFrame**⁶ eller en **JApplet**⁷, alt efter hvordan programmet er blevet aktiveret. Startes programmet som applet, oprettes der en **MainApplet**-klasse, mens der oprettes en **MainWindow**-klasse hvis spillet eksekveres på normal vis.

Til vinduet tilføjes der nu spilleplade, maskot, hjælpe tekst, kontrolknapper og menulinje, hvorefter "Nyt spil"-skærmen vises. Når brugeren har valgt en sværhedsgrad genereres der en sudoku af den valgte sværhedsgrad, hvorefter spillepladen aktiveres og den primære programløkke initialiseres. Det er her, det væsentligste i Figur 6 findes.

Her tjekkes der ved hver "iteration"⁸ om sudokuen er løst. Hvis det er tilfældet, vises "Tillykke"-skærmen, hvor brugeren får vist statistik og har mulighed for at starte et nyt spil.

Hvis sudokuen ikke er løst, har brugeren mulighed for at placere et tal eller bede om hjælp. Placeres der et tal på spillepladen starter "løkken" forfra, idet der igen tjekkes om sudokuen er løst.

⁶En **JFrame** er hovedvinduet i en Java applikation.

⁷En **JApplet** er "hovedvinduet" i en applet, der kan vises i internetbrowsere.

⁸Der er som sådan ikke tale om iterationer, da "løkken" blot genstarter hver gang brugeren har foretaget et valg og ikke fast, som i f.eks. en **for**- eller **while**-løkke.

Hvis brugeren beder om assistance i stedet for at placere et tal på pladen aktiveres hjælpefunktionen. Den starter med at tjekke om der er nogle forkert placerede tal. Hvis det er tilfældet markeres felterne med rød og maskotten fortæller brugeren at der er noget galt. Er der ingen fejl kaldes sudokuløseren, der finder et felt, som kan løses udelukkende ved hjælp af de basale sudokuregler. Er der fejl på spillepladen vises der ikke hjælp til løsbare felter. Hvis brugeren vil have vist hjælp til løsbare felter må de fjerne de forkerte tal først. Er der i forvejen markeret et løsbart felt fjernes den gamle markering når en ny vises. Når hjælpefunktionen er færdig afventer programmet interaktion fra brugeren igen, altså uden at tjekke om sudokuen er færdig, eftersom der ikke er ændret i spillepladen.

Når en ny sudoku skal genereres dannes der et nyt statistikobjekt og spillebrættet opdateres, dog uden at der oprettes et nyt objekt. Den grafiske repræsentation ændres også blot så den viser det nye spillebræt. Vi valgte ikke at lave nye objekter til den grafiske repræsentation, da vi alligevel havde besluttet os for ikke at bruge andre spilstørrelser end 9x9 og derfor var det nemmere blot at ændre det bagvedliggende data uden at skulle initialisere størstedelen af programmets objekter igen.

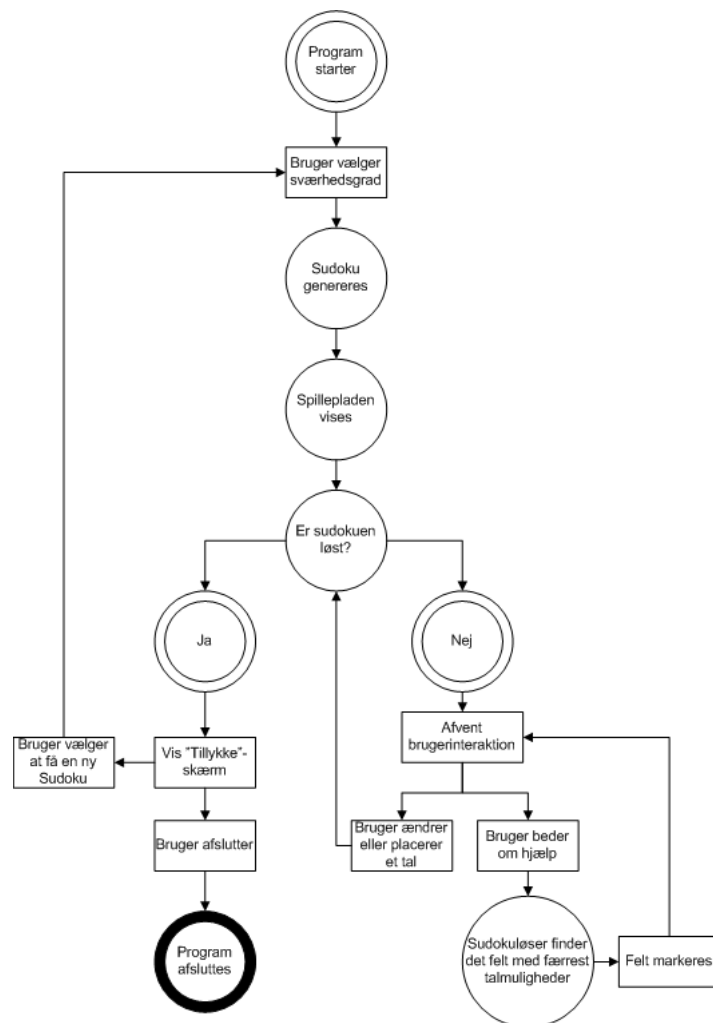
8.2 Logik i Control

Control indeholder de primære indgangspunkter for programmet, hvadenten det startes som applet eller applikation. Desuden er der også forskellige **Actions**, der bliver sammenkædet med knapperne og menupunkterne i brugergrænsefladen.

De vigtigste **Actions** i **Control** indbefatter: **HelpAction**, der tager sig af at vise fejl på spillepladen eller starte sudokuløseren for at finde et løsbart felt, der efterfølgende markeres. **DifficultyAction**, der viser skærmen hvorfra der kan vælges et nyt spil. **DifficultyAction** tager sig også af at sørge for at der bliver genereret en ny sudoku samt at spillepladen bliver gentegnet og at statistikken bliver nulstillet. **NumberAction** viser en dialog hvorfra der kan vælges et tal at placere på spillepladen. Dialogen giver også mulighed for at fjerne tal igen.

8.3 Sudokuløser

Sudokuløseren består af flere dele. Den del der bliver kaldt direkte, **solveField**, som også fungerer som guide til resten af løseren. En **solverLevelOne** og en **solverLevelTwo**. Hver af disse løserniveauer kan tage en løsning et niveau dybere. Den første kan kun kigge på feltet selv og finde en løsning til det ved at kigge på de felter der umiddelbart er i samme kvadrant, kolonne og række. **solverLevelTwo** vil kigge på om det givne felt er det eneste sted hvori der kan stå et tal, i forhold til rækken, kolonnen og kvadranten, hvis dette sted er det eneste bliver det returneret.



Figur 6: Flowdiagram over programforløb

8.3.1 solveField

Kalder `solverLevelOne`, gemmer et resultatarray indeholdende de værdier der kom ud af den sudokuløser, det vil sige, de mulige værdier for en endelig løsning.

Derefter, hvis der kun er en mulig løsning tilbage fra `solverLevelOne` bliver dette returneret til den funktion `solveField` blev kaldt fra. Hvis ikke den løsning er unik vil resultat arrayet blive sendt videre til `solverLevelTwo`. Resultatet herfra bliver derefter returneret.

8.3.2 solverLevelOne

Denne løserfunktion får inddata som et array af mulige resultater for sit eget felt, positionen af feltet på spillepladen og spillepladen selv. Den vil derefter kopiere arrayet af mulige værdier til sit eget locale array og så systematisk sætte de værdier til 0, som feltet ikke kan have som værdi. Det vil sige, tjekke de værdier feltet har som naboer i dets kvadrant, kolonne og række. Hver af disse værdier er ikke mulige løsninger og bliver derfor fjernet fra arrayet af mulige løsninger.

Når der ikke er flere felter at tjekke slutter funktionen og sender det nye array af mulige værdier tilbage.

8.3.3 solverLevelTwo

Løseren her er noget mere avanceret end den foregående. Den får de samme inputs, men skal bearbejde dem meget forskelligt fra den foregående metode.

Her er der allerede tjekket på den måde som `solverLevelOne` gør det for feltet selv, og de værdier er ikke inkluderet i de mulige værdier. Der bliver lavet tre kopier af dette array til brug for de tre lidt forskellige algoritmer der kommer efter. Den første algoritme tjekker alle de andre felter igennem som findes i samme række som feltet der skal løses, finder deres position en ad gangen. Derefter køres `solverLevelOne` på alle disse felter, de mulige løsninger de har bliver fjernet fra de mulige løsninger for det felt der skal løses. Hvis der efter en endt operation her kun er én løsning tilbage, det vil sige at det givne felt er det eneste felt i dens række som kan indeholde et bestemt tal vil det være løsningen på feltet. Dette gælder fordi der skal være præcis én af hver værdi i hver række, så hvis der ikke er andre muligheder for at det tal kan være der skal det være på den plads. Derfor vil dette tal blive returneret til `solveField`. Den næste algoritme vil tjekke kolonnen tilhørende til det enkelte felt, og så ellers gå efter samme princip som med den foregående.

Også den tredje algoritme gør det samme, dog er algoritmen for at finde start positionerne for de andre felter i samme kvadrant noget mere avanceret end dem for at finde positioner for række og kolonne felter.

Funktionen tjekker altså om der er en mulig værdier, der kun er at finde på ét felt i en række, kvadrant og kolonne, selvom alle felterne måtte have flere forskellige muligheder.

8.4 Sudokumatematik

Som hjælpeklasse til Sudokuløseren oprettede vi klassen **SudokuMath**. Funktionskaldene fra denne klasse kan udfra spillets indstillinger samt Id-nummeret på et givent felt udregne række-, kolonne- og kvadrant-nummer. Derudover kan den ligeledes returnere et array med den givne række, kolonne eller kvadrant. I vores færdige implementation bruger vi i programmet kun standard 9x9 Sudoku, men **SudokuMath** er skrevet således at hvis implementationen af Sudokuer på andre størrelser (f.eks. 4x4 eller 16x16) er ønsket, vil matematikken blive korrekt beregnet. Løseren, der som sagt kun bruger data fra **SudokuMath**, vil således blot kunne bede om de tal eller numre den skal bruge, uden at bekymre sig om den egentlige størrelse af brættet.

Grunden til at vi valgte at anbringe metoderne i en klasse for sig selv, var at gøre løseren mere overskuelig at arbejde med, læse, og forstå. Matematikken er i nogen af tilfældene ikke helt så simpel, og det hjælper meget at man ved hjælp af et enkelt funktionskald med sikkerhed kan få eksempelvis rækkenummeret vist, istedet for at skulle indsætte udregningen til hvorledes det gøres. Derudover undgår vi ligeledes også problemet med at der indgår såkaldte "magiske tal" i løseren.

8.5 Hjælpefunktionen

Hjælpefunktionen er en funktion der alene eksisterer på grund af den givne opgave beskrivelse. Da programmet skal bruges af forholdsvis små børn og de skal have mulighed for at få hjælp og spille spillet uden en udefrakommende voksen eller ældre til at forklare eller få dem videre.

Hjælpeknapen er altid tilgængelig medens spillet er i gang. Når brugeren trykker på den starter hjælpefunktionen, der som standard ser hele det genererede bræt igennem og så stiller værdierne, som brugeren har intastet, op imod det korrekt løste bræt. Hvis de indtastede værdier ikke stemmer overens med den korrekte løsning markeres disse felter som fejl, og farves røde. På den måde kan brugeren se hvor der er fejl, og derefter rette dem. Hvis der ikke er nogen fejl vil funktionen gå ind og få løseren til at finde et tilfældigt løsbart felt. Dette felt bliver markeret med grøn, og brugeren bliver gjort opmærksomt på at feltet der er fremhævet kun har én løsning.

Selvfølgelig er det muligt at nogen af de fremhævede felter er sværere at løse end andre, men hvis ikke vi bruger en så forholdsvis stærk algoritme, vil spillet heller ikke være udfordrende for de smarteste af brugerne. Det er også muligt at misbruge hjælperen, på den bekostning at man får en dårligere statistik til slut. Det vil dog altid være muligt at udlede et markeret felts

værdi ud fra principperne om de basale sudokuregler for talplaceringer, samt gensidig udelukkelse⁹.

8.6 Generator

Sudokugeneratoren bruger sudokuløseren til at generere nye sudokuer. Der startes med et udfyldt sudoku bræt der bliver vilkårligt blandet, dog følgende basale regler for hvilke operationer man kan udføre på en sudokuplade uden at invalidere den:

- Man kan bytte om på "kvadrantkolonner"¹⁰.
- Man kan bytte om på "kvadrantrækker"¹¹.
- Man kan bytte om på kolonner inden for kvadrantkolonner.
- Man kan bytte om på rækker inden for kvadrantrækker.

Ovenstående regler vil aldrig invalidere en sudokuplade, da der hvis de overholdes altid vil være tallene 1-9 at finde i hver kvadrant, række samt kolonne.

Derefter fjerner generatoren et tilfældigt felt på brættet og kontrollerer med løseren at det kan placeres igen ved hjælp af de resterende felter på sudokupladen. Hvis feltet ikke kan placeres sættes det tilbage og der prøves med det efterfølgende felt. Sådan fortsættes indtil et felt kan fjernes eller at der ikke kan fjernes flere felter overhovedet, hvor generatorfunktionen da vil afbryde. Hvis feltet kunne placeres igen, forbliver det fjernet fra brættet og et nyt tilfældigt felt vælges. Processen gentages indtil det ønskede antal felter, som bliver hentet fra spillets indstillinger¹², er fjernet eller det ikke er muligt at fjerne flere felter.

9 Funktionstest

Funktionstestene kontrollerer at vores program producerer de forventede ud-data ud fra specificeret inddata. Funktionstestene er til for at teste nogle af de centrale dele i vores program. Alle funktionstest og deres resultater kan ses i bilag "Test specifikation" på side 29.

⁹Gensidig udelukkelse vil sige at hvis et felt har flere mulige værdier, men nogle af værdierne kun kan placeres andetsteds i kvadranten, kolonnen eller rækken, skal de naturligvis ikke placeres på det pågældende felt.

¹⁰En kvadrantkolonne består af en kolonne bestående af kvadranter. I en 9x9-sudoku vil det sige at kolonnerne 1-3, 4-6 og 7-9 er hhv. kvadrantkolonne 1, 2 og 3.

¹¹En kvadrantrække er som en kvadrantkolonne, blot med rækker i stedet for kolonner. Deres placering i en 9x9-Sudoku er de samme.

¹²Findes i klassen `GameSettings`.

9.1 Unittest

Unittestene er lavet for at teste vigtige moduler i programmet virker som forventet. De tester blandt andet klassen `SudokuMath` som er matematikken der bruges til at finde felter på spillepladen. Der er også lavet Unittest til `Helper` klassen som bruges til at give hjælp til brugeren undervejs i spillet.

Testene af matematikfunktionerne kontrollerede hvorvidt de essentielle hjælpefunktioner til at få "sudokukoordinater"¹³ ud fra et Id-nummer på et felt. Endvidere blev det kontrolleret om andre hjælpefunktioner kunne returnere alle Id-numre på felter i et givent felts række, kolonne og kvadrant.

Heldigvis viste ingen af disse test fejl, da fejl ville have medført at store dele af den grundlæggende generator- og løserlogik skulle være skrevet om.

Hjælpefunktionen blev udsat for kontrol af om den var i stand til korrekt at finde ud af om felter var løsbare eller ej. En helt tom spilleplade ville derfor ikke have nogen løsbare felter, da vores sudokuløser ikke var programmeret til at gætte, mens en spilleplade hvor en af kvadranterne var fyldt ud med f.eks. otte ud af de ni gyldige tal¹⁴ ville returnere Id-nummeret på det felt hvor det niende tal skulle stå.

Testene af hjælpefunktionen viste heller ingen fejl.

9.2 Blackbox

Da vores sudokugenerator skulle testes var der kun en umiddelbar fremgangsmåde. Da sudokuerne bliver genereret ved hjælp af vores egen løser, kunne denne ikke bruges til at test af sudokuernes korrekthed. Derfor måtte vi kontrollere vores genererede sudokuer ved hjælp af en anden sudokuløser som vi stolede på var korrekt. Den løser mange sudokusider henviser til, befinder sig på <http://sudoku.sf.net>. Vi fik vores program til at printe sudokuer ud i et format der kunne kopieres ind i den anden sudokuløser manuelt. Vi fik derefter genereret et stort antal sudokuer og testede deres singularitet¹⁵. Vi konstaterede at der var et antal sudokuer der ikke var singulære, men havde flere løsninger. Dette var et problem da en bruger dermed kunne løse en sudoku helt korrekt, men vores program ville ikke anerkende løsningen, da den ikke ville stemme overens med den, der lå gemt i programmet.

Efter at have konstateret at der var en fejl i vores algoritme til at løse sudokuer (og derved fejl når vi genererer sudokuer) blev algoritmen rettet og testet igen. Vi testede med et større antal af den sværhedsgrad, der i forrige test havde genereret flest fejl, men denne gang fandt vi ingen problemer, hvormed vi kunne opfylde vores krav om singulære, løsbare sudokuer.

¹³ Kolonnetal, rækketal og kvadrantal.

¹⁴ På en 9x9-spilleplade er der altså tallene 1-9.

¹⁵ At en sudoku er singulær betyder at den kun har én løsning. En singulær sudoku kaldes også "ægte".

10 Brugertest

Bruger testene er til for at sikre sig at programmet er imødekommende og behager brugeren, samt at det opfylder de krav der skulle være fra brugerens side. Der blev lavet tre forskellige brugertest. Den ene tester tilgængelighed, tekst, grænseflade og motivation i programmet. Den anden er en test af linuxkompatibilitet. Den tredje testede applet- og browserkompatibilitet. Alle brugertest og deres resultater kan ses i bilag "Test specifikation" på side 29

10.1 Tilgængelighed, tekst, grænseflade og motivation

Vores program er designet til undervisningsbrug i 1. - 3. klasse. Vi har derfor testet vores program på 6 elever fra 2.B på Dronninggårdsskolen i Holte. De blev under testen overladt til dem selv uden instruktioner. Børnene blev så observeret mens de brugte programmet. Der blev taget notater undervejs og da første omgang var afsluttet blev de diskuteret. Programmet blev rettet til efter resultaterne af første afprøvning og børnene blev præsenteret for den opdaterede udgave af programmet. Efter anden omgang som forløb som første gav børnene deres feedback. Et af vores hovedpunkter i feedbacken var hvor vidt børnene kunne lide det grafiske layout. Da vi selv har tegnet det hele, for ikke at bruge noget materiale beskyttet af ophavsret, var der uvished om hvorvidt vores maskot var tiltalende for børn. Heldigvis var brugerne meget positive overfor både vores maskot, fåret Dolly, og det resterende grafiske layout, som de også fandt let at bruge.

Testnavn	Beskrivelse	Succeskriterie	Testmetode	Opnået resultat	Test opfyldt?	Kommentarer
Tilgængelighed 1	Kan forsøgspersonerne selv kunne starte et Sudokuspil?	6/6 skal selv kunne starte et Sudokuspil.	Observation af forsøgspersoner.	6/6	Ja	Alle børn kunne selv med det samme finde ud af at starte et spil.
Tilgængelighed 2	Kan forsøgspersonerne selv placere tal på Sudokuplacen?	6/6 skal uden ekstern hjælp kunne placere tal på Sudokupladen.	Observation af forsøgspersoner.	6/6	Ja	Alle børn kunne selv med det samme finde ud af at placere tal på Sudokupladen.
Tilgængelighed 3	Kan forsøgspersonerne selv løse en valgfri Sudoku uden forklaring?	5/6 skal uden ekstern hjælp kunne løse en valgfri (let, mellem eller svær) Sudoku uden forklaring.	Observation af forsøgspersoner.	6/6	Ja	Alle børnene kunne som minimum løse en let Sudoku.
Tilgængelighed 4	Kan forsøgspersonerne selv afslutte programmet?	5/6 skal uden ekstern hjælp kunne afslutte programmet.	Observation af forsøgspersoner.	6/6	Ja	Alle børnene brugte X'et i vinduets øverste højre hjørne. Ingen brugte menuen.
Tilgængelighed 5	Kan forsøgspersonerne bede om at få hjælp?	6/6 skal uden ekstern hjælp bede om at få hjælp i programmet.	Observation af forsøgspersoner.	2/6	Nej	Enkelte aktiverede hjælpen, men fårets respons på børnenes talvalg blev (groft) udnyttet i stedet.
Tilgængelighed 6	Kan forsøgspersonerne udlede hvilken værdi et felt skal have ud fra den i programmet indbyggede hjælp?	5/6 skal uden ekstern hjælp ved en let Sudoku kunne udlede hvilken værdi et felt skal have.	Observation af forsøgspersoner.	0/6	Nej	Hjælpfunktionen blev brugt meget sparsomt og det markerede felt blev ikke efterfølgende løst.
Sværhedsgrad 1	Kan forsøgspersonerne uden ekstern hjælp løse en let Sudoku?	6/6 skal uden ekstern hjælp kunne løse en let Sudoku på 10 minutter.	Observation af forsøgspersoner.	6/6	Ja	
Sværhedsgrad 2	Kan forsøgspersonerne uden ekstern hjælp løse en middel Sudoku?	4/6 skal uden ekstern hjælp kunne løse en mellem Sudoku på 20 minutter.	Observation af forsøgspersoner.	6/6	Ja	

Tabel 1: Resultater fra brugertest af grænseflade og funktionalitet

Testnavn	Beskrivelse	Succeskriterie	Testmetode	Opnået resultat	Test opfyldt?	Kommentarer
Sværhedsgrad 3	Kan forsøgspersonerne uden ekstern hjælp løse en svær Sudoku?	2/6 skal uden ekstern hjælp kunne løse en svær Sudoku på 20 minutter.	Observation af forsøgspersoner.	2/6	Ja	Ét sted blev der samarbejdet (løsning på 16 minutter) og én pige klarede Sudokuen på 20 minutter.
Grænseflade 1	Finder forsøgspersonerne grænsefladen indbydende?	5/6 skal finde grænsefladen indbydende.	Interview af forsøgspersoner.	6/6	Ja	Børnene kunne godt lide grænsefladens udseende.
Motivation 1	Synes forsøgspersonerne om programmets maskot?	5/6 skal synes om programmets maskot.	Interview og observation af forsøgspersoner.	6/6	Ja	De synes maskotten var sød og de nød det den sagde.
Motivation 2	Finder forsøgspersonerne statistikken brugbar og/eller motiverende?	5/6 skal finde statistikken motiverende.	Interview og observation af forsøgspersoner.	6/6	Ja	Det var et godt konkurrenceelement, plus at de kunne se om de blev bedre.
Tekst 1	Kan forsøgspersonerne forstå meningen med al teksten i programmet?	5/6 skal forstå meningen med al teksten i programmet.	Interview og observation af forsøgspersoner.	6/6	Ja	Der var ingen problemer med at forstå teksten i programmets mening.

Tabel 1: Resultater fra brugertest af grænseflade og funktionalitet

Tabel 10.1 viser alle vores testresultater fra første omgang af brugertestene. Som det kan ses var størstedelen af vores test vellykkede, og børnene gav udtryk for at størstedelen af vores program var intuitivt at bruge.

Ét punkt fejlede dog: Hjælpefunktionens udformning. Mange børn gjorde slet ikke brug af hjælpefunktionen, idet vi oprindeligt havde gjort maskotten i stand til at reagere positivt eller negativt på brugernes talvalg. Dermed var der ikke behov for børnene for at bruge hjælpefunktionen, og ej heller behov for selv at løse Sudokuen, da vi var kommet til at give en genvej til hurtigt at løse Sudokuspillene. Endvidere så de få der brugte hjælpefunktionen ikke ud til at forstå de med grønt markerede felters betydning. Meningen med de røde feltmarkeringer der indikerede fejl, var dog klar nok og børnene blev med det samme opmærksomme på at der var noget galt.

Efter første testsession ændrede vi maskottens opførsel, så den ikke længere reagerede positivt eller negativt på talvalg, men blot henviste til hjælpefunktionen, hvis ikke den skulle vise hjælp eller gøre brugeren opmærksom på fejl. Derudover ændrede vi teksten når hjælpen blev brugt, så det klart fremgik at feltet kun havde én mulig værdi. Under anden testsession blev børnenes handlingsmønstre ændret gevaldigt, idet de ikke længere kunne misbruge fårest reaktion på deres talvalg til at afgøre om et felt var udfyldt korrekt eller ej. Børnene begyndte i stedet at bruge hjælpefunktionen til at kontrollere om de havde lavet fejl og også til at få vink til videre fremgang, hvis de sad *helt* fast i en Sudoku. Dette handlingsmønster stemte mere overens med vores oprindelige intentioner, og overdreven brug af hjælpefunktionen viste sig da også på statistikoversigten, hvormed “Antal fejl” og “Antal hjælp” også kunne indgå i børnenes individuelle opfattelse af deres præstation, samt deres naturlige sammenligning med kammeraternes løsningsstatistikker.

Sværhedsgraderne viste sig også at være passende for målgruppens faglige niveau. Alle børnene kunne løse en “Let” Sudoku inden for 10 minutter. Størstedelen var også i stand til at løse Sudokuer på “Mellem”-niveauet, omend det naturligvis tog lidt længere tid at løse pladerne. Den højeste sværhedsgrad udfordrede heldigvis børnene, og kun to steder blev den løst på under 20 minutter, hvormed vores intention med sværhedsgraden var indfriet.

10.2 Linuxkompabilitet

Da vi ikke havde en ordentlig linux maskine i vores gruppe måtte vi låne en. Søren Houen fra Gruppe otte havde installeret Ubuntu 6.06 (kernel 2.18) på sin bærbare computer og udlånte den venligst som testcomputer til denne test.

Vi testede at følgende funktionalitet var mulig:

- Placering og fjernelse af tal på spillepladen.
- Start af Sudoku med alle tre sværhedsgrader.

- Løsning af Sudokuer.
- Visning af statistik.
- Brug af menu til at starte nyt spil samt afslutte programmet.
- Markering af løsbart felt vha. hjælpefunktionen.
- Markering af forkert placerede felter vha. hjælpefunktionen.

I henhold til forventningerne om bevarelse af funktionalitet på tværs af platforme ved brug af Java-plattformen, blev der ikke opdaget nogen problemer i de test vi udførte.

10.3 Appletkompatibilitet

Jævnfør vores kravspecifikation, skulle vores program fungere som en Java Applet i de to internetbrowsere Internet Explorer (version 5.5 og derover) samt Mozilla Firefox (version 1.5 og derover).

Vi udførte derfor test, i stil med dem i afsnittet om Linuxkompatibilitet, der kontrollerede hvorvidt den nødvendige funktionalitet var til stede.

Også i henhold til forventningerne om bevarelse af funktionalitet ved brug som Java Applet, blev der heller ikke i denne test opdaget nogen problemer.

11 Konklusioner

Vi blev stillet en opgave om at kreere et Sudokuprogram, der kunne bruges i undervisningsøjemed i folkeskolen. Målgruppen var børn i 1.-3. klasse, dvs. i aldersgruppen 6-10 år. Sudokuprogrammet skulle være et alternativ til den nuværende anvendelse af Sudoku i undervisningen, hvor børnene får udleveret og skal løse opgaverne på papir.

Denne løsning har et par ulemper, bl.a. at børnene ikke kan få hjælp undervejs medmindre der er en voksen i nærheden, der så skal til at sætte sig ind i barnets nuværende løsning. Derudover kan børnene heller ikke nemt blive gjort opmærksom på, hvis de har lavet fejl, hvorfor de kan blive ledt ud på en lang sti, for til sidst måske enten at have løst Sudokuen forkert eller ende i en blindgyde hvorfra de ikke kan komme videre. Desuden er børnene begrænset af det antal Sudokuer de får udleveret i skolen. Børnene kan også have et problem med sværhedsgraden, hvis de udleverede Sudokuer ikke passer til deres niveau, hvilket heller ikke er ideelt, da de mest fremmelige børn potentielt vil kede sig mens de svage børn kan komme ud for ikke at kunne løse opgaverne.

Disse problemer finder man ikke i vores Sudoku-program. Programmet tillader børnene at løse så mange Sudokuer som de gider, da spillepladerne bliver genereret tilfældigt. Derudover er der graduerbare sværhedsgrader, som gerne skulle tilbyde udfordring for de fleste brugere, og ikke kun

“standard”-brugerne. De forskellige sværhedsgrader tillader også progression, skulle børnene føle sig klar til sværere opgaver.

Derudover står børnene heller ikke over for problemet med at skulle have en voksen til stede for at kunne hjælpe dem med at finde fejl eller give vink om i hvilken retning det ville være hensigtsmæssigt at fortsætte, da den i vores program bagvedliggende Sudokuløser meget hurtigt kan vise hvor der er fejl og samtidig kan vise, hvilke felter der kan løses hvis man kender Sudokureglerne.

I vores projekt har vi lagt stor vægt på - og stort arbejde i - programmets grafiske fremtoning. For at appellere til målgruppen gjorde vi stor brug af pastelfarver og store, klare grænsefladeelementer. Vi valgte også selv at kreere vores maskot for at frihed til at bruge den som vi lystede, uden at løbe ind i problemer med ophavsret.

Under udviklingen gjorde vi meget ud af at gøre programkoden så let at udvide som muligt. Derfor valgte vi at separere programlogik, data og udseende. Vi forsøgte derudover at gøre så stor brug af abstrakte klasser og **Interfaces** i Java som muligt, for at skabe en “plan” som programmets moduler kunne interagere efter. På den måde er det let at tilføje ekstra funktionalitet til spillet - blot det implementerer de af **Interface**ne påkrævede funktioner, burde nye moduler kunne implementeres i spillet uden nævneværdige modifikationer.

For at gøre programmet godt at bruge var et vigtigt krav for os at det var responsivt og at man ikke skulle vente på data. Dette var en af grundene til at vi valgte at gøre brug af primitive datastrukturer, da operationer på dem kan udføres hurtigt og simpelt. På den måde kunne vi sørge for at der ikke er nogen mærkbar ventetid når brugeren beder om at få genereret en ny Sudoku.

Det vigtige i et program der henvender sig til den fastsatte målgruppe er, at programmet er let og intuitivt at bruge - for *målgruppen*, ikke for udviklere, interessenter o.l. - samt at det er en indbydende og “venlig” brugergrænseflade. Da dette ikke er noget der kan afgøres ved hjælp af funktionstest, er et meget vigtigt aspekt af programudvikling til den pågældende målgruppe at der bliver udført brugertest, fordi det kun er målgruppen der kan afgøre om programmet overordnet set er vellykket og ordentligt udført.

Vores brugertest med børn viste generel tilfredshed med programmets funktionalitet og fremtoning. Hjælpefunktionen viste sig at indeholde nogle svagheder, men på baggrund af den feedback vi modtog var det ikke det store problem at rette væsentligt op på det. Samlet set var testgruppen til slut positive over for vores program og ytrede, at de “meget gerne ville bruge det [programmet] i skolen i stedet for papirerne”.

Funktionstest kan dog naturligvis ikke undværes, og vi fandt da også fejl i vores generatoralgoritme. Men stillet overfor de specifikke test der fejlede, fandt vi fejlen og kunne efterfølgende ikke konstatere flere problemer i algoritmen.

12 Bilag

12.1 Testspecifikation

12.2 Designdokument