Sudoku til undervisningsbrug

Emil Erik Hansen Julian Møller Klaes Bo Rasmussen
Steen Nordsmark Pedersen

11. juni 2007

Indhold

1 Formalia

Projekttitel	Sudoku til undervisningsbrug
Gruppenr.	4
Deltagere	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 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 syntes at lige præcis dette emne var det vi ville beskæftige os med. Det lød spændende at kunne beskæftige sig med mange forskellige aspekter af software-udvikling , specielt at skabe en pæn, underholdende og overskuelig brugergrænseflade, HCI , arbejde med matematikken og algoritmerne bag en Sudoku og om vi kunne sætte sværhedsgraden således at de der kunne være noget hele målgruppen kunne finde ud af. 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 [indsæt mere om nødvendigt, ved det ikke lige i dette øjeblik].

Sudoku bliver brugt i folkeskolen i undervisningen til at lære børnene matematik. Børnene får udleveret Sudokuer på papir, som de skal løse, enten i

klassen eller som hjemmearbejde. Denne metode har visse ulemper. Børnene kan dels ikke få hjælp hvis de sidder fast. 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 både for 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. Hvis børnene sættes til at løse Sudokuer på computere, bliver de derudover også vant til at bruge IT i en tidlig alder - et kendskab man ikke kan være foruden nu til dags.

4 Problemformulering og afgrænsning

4.1 Baggrund

En Sudoku er en slags spilleplade med "som regel" 9 gange 9 felter. Disse felter er når spillet er færdigt udfyldt med tallene fra 1 til 9. Der er rækker og kolonner og kvardranter i denne spilleplade, hver af disse er alle på 9 felter, og der er 9 af hver af dem på en fuld "normal" spilleplade. Rækker og kolonner er ikke så svære at skælne, men kvardranterne 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 kvardranter.

I hver række, kolonne og kvardrant skal der være præcis 1 af hver af de 9 tal. Med denne viden skal man kunne udfylde en sudoku hvor nogle eller flere af felterne er blanke. Jo færre tal der er på pladen, jo sværere bliver sudokuen at løse og man skal tænke meget langt frem for at kunne finde ud af hvordan man skal løse den. Det specielle ved at generere sådan en sudoku er at det ikke er en rigtig sudoku hvis man kommer til at fjerne nogen tal fra pladen der gør at der er flere end 1 mulighed for at løse den. Hvis man begynder med at have 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 plade 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 han/hun sidder fast. Dette skyldes det basale krav om at hjælp skal være muligt at få uden at skulle spørge en udefrakommende. Man skal ligeledes kunne få af vide når spillet er færdigt, få et nyt spil på et hver tidspunkt og afslutte programmet når man har lyst. Programmet skal virke indbyddende og sjovt samt let

tilgængeligt for mindst den givne brugergruppe.

4.2 Problemformulering

Vi har valgt kun at kunne generere 1 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 sudoku plade og derefter fjerner vi et tal af gangen. Efter at fjerne et tal beder vi vores sudoku løser om at løse det givne felt, hvis den kan lader vi feltet være tomt, derefter 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 med, da der altid vil være mindst 1 felt denne løser kan løse.

5 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.

Kravene fra interessenten, samt de krav vi selv har tilføjet blev delt op i katorierne SUDOKU, BRUGBARHED, MOTIVATION og TEKNIK.

5.1 Kategorien SUDOKU

Indeholder de første tre af kravene, direkte konverteret ind til kravspecifikationen. Denne kategori indeholder de egentlige spiltekniske krav af selve sudoku-spillet. 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.

5.2 Kategorien BRUGBARHED

Indeholder punkterne der omhandler anvendeligheden og kravene læsefærdighed. Som et krav syntes vi at der ved lettilgængelighed mentes at brugeren skulle kunne komme igang med at benytte programmet indenfor et meget kort stykke tid, hvilket sætter et stort krav til at vores grafiske brugergrænseflade er overskuelig og ligetil.

Læsefærdighedskravene valgte vi at definere som at teksten ikke bør have en LIX-værdi over 8, men ville være iorden, så længe børnene forstod den tekst der blev skrevet. Derudover valgte vi at sætte det som et krav at brugeren skulle kunne manurere sig rundt i programmet udelukkende ved brug af musen. Dette syntes vi var passende for at børnene kan holde fokus på spillet, uden at skulle kigge ned på tastaturet for at kigge efter knapper. I standard Sudoku er øjnene ligeledes antid på spillepladen. Disse 2 krav er sat som subkrav til anvendeligheden, da vi mente at de var nødvendige for at opnå en letanvendelighed for vores målgruppe. Vi valgte ligeledes at tilføje et selvstændigt krav for at det skulle være muligt at slette de tal man sætter, i tilfælde af at disse er forkerte, ligesom man i en normal Sudoku man viske sine tal ud, hvis man har lavet en fejl.

5.3 Kategorien MOTIVATION

Indeholder punkterne der omhandler at programmet skal være sjovt og underholdende. Her er det originale punkt delt op i 2 mere konsekvente punkter, da vi fandt at beskrivelsen af det originale krav var for løst. De 2 nye punkter er at der skal være en maskot der følger brugeren gennem programmet og eventuelt bruges til at videresende beskeder fra programmet til brugeren. Med denne maskot har det selvfølgelig været meningen at den skal kunne være et som børnene skal finde sødt, og passe ind i den resterende del af brugergrænsefladen.

5.4 Kategorien TEKNIK

Indeholder kravene der omhandler de tekniske krav til computeren som programmet skal køres på. Vi har her valgt at sætte version 1.5 java som et krav istedet for den nyeste version. Grunden til dette valg er at java 1.5 er en mere udbredt standard, og vi ikke vil have at programmets krav gør det svært. [ikke færdigt]

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.

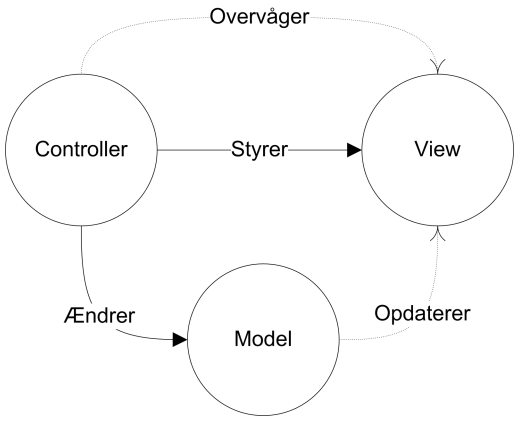
Der er flere hovedpunkter designmæssigt. Det ene og det vigtigste er græsefladen. Da det er et program for børn skal den være simpel og let at bruge. Derudover skal den være tillokkende og flot for at holde børnenes interesse.

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.

Endvidere skal systemet være let at udvide med ny funktionalitet, hvilket er en af grundende 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.

6.2 MVC

Vores design er bygget op omkring den klassiske MVC (Model-View-Control) 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 seperat del, komplet adskilt fra programmets udseende og data. Derudover kan man nemt f.eks. skifte spillets "motor" - igen uden at skulle ændre hverken programlogikken eller den grafiske grænseflade. Det fulde design dokument uddyber designet og kan findes som bilag på side 14. 1 viser det overordnede samspil mellem de tre dele.



Figur 1: 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-spille plader 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.

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 skærmbilleder og så der bliver udført handlinger

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 løsbar Sudoku hvorefter det kalder `View` modulet til at lave en grafisk repræsentation som brugeren så 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.

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. På den måde 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 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.

7 Grafisk design

Det grafiske design er en væsentlig den 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 bruger test, men det er muligt at gøre tingene simple, men flotte før det er muligt at teste det. Vores brugergrænseflade består af tre skærmbilleder "Vælg Sudoku", "Sudoku" og "Tillykke".

7.0.1 Vælg Sudoku

På dette skærmbillede har brugeren valget mellem en "Let", "Mellem" og "Svær" Sudoku. Når en af disse er valgt fortsættes der til næste skærmbillede.

Figur 2 viser hvordan dette skærmbillede ser ud.

Figur 2: Skærmen hvor brugeren kan vælge en sværhedsgrad

7.0.2 Sudoku

Sudoku skærmbilledet består af en menulinje, en Sudoku spille plade og to knapper. Fra menulinjen er det muligt at starte et nyt spil, lukke spillet, få hjælp til at læse Sudokuen, og at få information om hvor de generelle Sudoku regler kan findes. På Sudoku pladen 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 skærmbillede og derved starte en ny Sudoku.

Figur 3 viser hvordan dette skærmbillede ser ud.

7.0.3 Tillykke

Det sidste skærmbilledet er "Tillykke" billedet. Der vises statistik for den løste Sudoku samt en knap til at starte et nyt spil.

Figur 4 viser hvordan dette skærmbillede ser ud.

Figur 3: Sudoku spillepladen

Figur 4: Tillykke-skærmen der vises når en Sudoku er løst

8 Teknisk beskrivelse

9 Afprøvning

Vores Sudoku program kræver to former for test. Funktions test og bruger test. Funktions testene kontrollerer at vores program producerer det forventede output ud fra det givne input. 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. Alle test og deres resultater kan ses i bilag "Test specifikation" på side 14.

9.1 Funktions test

Funktions testene er til for at teste nogle af de centrale dele i vores program. Nogle er lavet med JUnit test og andre er blackboxtest.

9.1.1 JUnit

JUnit testene er lavet for at teste at centrale 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 JUnit

test til Helper klassen som bruges til at give hjælp til brugeren undervejs i spillet.

9.1.2 Blackbox

Da vores Sudoku generator skulle testes var der kun en umiddel bar fremgangsmåde. Da de bliver genereret ved hjælp af vores løser kunne denne ikke bruges til at test af Sudokuernes korrekhed. Derfor måtte vi kontrollere vores genererede Sudokuer ved hjælp af en anden Sudoku løser som vi stolede på var korrekt. Den løser de fleste Sudoku sider henviser til er den der befinder sig på http://Sudoku.sf.net. Vi fik vores program til at printe sine Sudokuer ud i et format der kunne kopieres ind i den anden Sudoku løser manuelt. 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.

9.2 Brugertest

Der blev lavet to forskellige brugertest. Den ene tester tilgængelighed, tekst, grænseflade og motivation i programmet. Den anden er en test af linuxkompatibilitet.

9.2.1 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 præsenteret for programmet og derefter 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.

9.2.2 Linuxkompabilitet

Da vi ikke havde en ordentlig linux maskine i vores gruppe måtte vi låne en. Søren Houen fra Gruppe 8 havde installeret Ubuntu 6.06 med kerne 2.18 og blev vores testcomputer til denne test.

9.3 Konklusion på test

Vi kan konkludere at bruger test i henhold til vores projekt er meget væsentlige. Da vores program kun er vellykket hvis børn i 1. - 3. klasse kan lide at bruge det er det derfor meget afgørende hvilket feedback vi får fra vores brugertest. Brugertestene var også væsentlige da de fandt fejl i programmet

der kunne have ledt til at en bruger havde siddet fast i en Sudoku. Alle vores test var vellykkede da nogle fandt fejl i programmet, andre steder der kunne forbedres og nogle dokumenterede at vores program virker efter intensionen.

10 Konklusioner

11 Bilag

11.1 Testspecifikation

11.2 Designdokument

11.3 Programkode