# Visualisering av data i en produktionsprocess

CARL ERIK HALMEID


KTH Numerisk analys
och datalogi

Examensarbete
Stockholm, Sverige 2004

TRITA-NA-E04105


Numerisk analys och datalogi
KTH
100 44 Stockholm

Department of Numerical Analysis
and Computer Science
Royal Institute of Technology
SE-100 44 Stockholm, Sweden

# Visualisering av data i en produktionsprocess

CARL ERIK HALMEID

TRITA-NA-E04105

Examensarbete i medieteknik om 20 poäng
vid Programmet för medieteknik,
Kungliga Tekniska Högskolan år 2004
Handledare var Frode Volden
Examinator var Nils Enlund

# Sammendrag

Den stadig økende konkurransen, ikke minst i prosessindustrien, fører til høyere krav til effektive produksjonsprosesser, muligheter for sporing og bedre kvalitetssikring.

Visualisering av data som blir generert i produksjonsprosessene vil være et godt hjelpemiddel for å gi operatører og analytikere hurtig og effektiv tilgang til data som er kritiske for produksjonskvaliteten.

Det eksisterer mange konsepter for visualisering av store og komplekse datamengder hvor hensikten er å vise mye data oversiktlig og på forholdsvis liten plass.

Denne oppgaven tar sikte på å gjennomgå ulike konsepter, og på bakgrunn av denne gjennomgangen designe en prototyp som kan benyttes for visualisering under gitte forutsetninger. Prototypen er designet slik at den kan utprøves på to konsepter. Den er så blitt tilpasset en reell produksjonsprosess for å teste om valgte konsepter kunne benyttes for en slik prosess.

Prototypen ble demonstrert for et lite utvalg brukere for å få tilbakemeldinger på konseptenes anvendbarhet, styrke og svakheter.

Konklusjonen er at idéene bak konseptene er gode, men krever visse visuelle tilpasninger samtidig som innføringen må skje gradvis sammen med god opplæring for at brukerne bedre skal forstå tankegangen og kunne utnytte konseptene optimalt.

# Abstract

That ever-growing competition, at least in the process industry, demands efficient production processes, prospects for tracking and quality assurance.

Visualizing the data that the production processes generate will be a good remedy for the operators and the analysts to quickly and efficient get access to data that are critical for the production quality.

Many concepts for visualizing large and complex data sets exist where the purpose is to visualize much data clearly on relatively small areas.

This thesis aim to examine different concepts and based on this examination, design a prototype that can be used for visualization under certain conditions. The prototype is designed in that way that it can test two concepts. It was then adapted to fit an actual production process to test if the chosen concepts can be used for that process.

The prototype was demonstrated for a small group of users in the purpose of getting feedback about the concepts usability, strengths and weaknesses.

The conclusion is that the ideas behind the concepts is good, but it requires some visual adjustments and at the same time the introduction should be done gradually and with good training to ensure that the users better should understand the concepts and use them optimally.

# Forord

Denne oppgaven er utført våren 2004 og er en del av en toårig påbygging til sivilingeniør i Elektronisk publisering og multimedieteknikk. Oppdragsgiver for denne oppgaven har vært Prediktor AS, og ScanWafer III har blitt benyttet som case study for å demonstrere prototypen på en reell produksjonsprosess.

Prediktor AS er lokalisert i Fredrikstad. De tilbyr komplette løsninger for optimalisering, sporbarhet og rapportering til prosessindustrien.

I tillegg har også IFE (Institutt For Energiteknikk) som er lokalisert i Halden, vært en god støttespiller i utarbeidelsen av prototypen. Instituttet driver med forskning blant annet innenfor kjernekrafts- og petroleumsindustrien og har utviklet gode brukergrensesnittkonsepter for industriproduksjonsovervåking (alarmhåndtering).

Jeg vil rette en stor takk til ansatte i Prediktor og spesielt veileder Erling Fledsberg for god oppfølging og tilrettelegging i forhold til ScanWafer III. I tillegg vil jeg takke Robin Welch og Øystein Veland ved IFE for konstruktive tilbakemeldinger både før og under arbeidet med prototypen.

Til slutt vil jeg takke Rune Hjelsvold og veileder Frode Volden, Høgskolen i Gjøvik, for gode råd og veiledning underveis i prosjektperioden.

# Innholdsfortegnelse

**1 INTRODUKSJON ..... 1**

1.1 EMNE OG OPPGAVEBESKRIVELSE ..... 1
1.2 MOTIVASJON ..... 2
1.3 UTFORDRINGER OG OPPGAVEBEGRENSNING ..... 2
1.4 METODE OG MÅL ..... 2
1.5 ORGANISERING AV RAPPORTEN ..... 3

**2 KONSEPTER FOR PRESENTASJON AV STORE OG KOMPLEKSE DATAMENGDER, TRESTRUKTURER OG PROSESSTOPOLOGIER ..... 4**

2.1 BAKGRUNN ..... 4
2.2 VISUALISERING ..... 5
2.2.1 Generelle problemer med visualiseringsteknikker ..... 5
2.2.2 Visualisering av trestrukturer og prosesstopologier ..... 6
2.2.3 Visualiseringsteknikker for store og komplekse datamengder ..... 12
2.2.3 Visualisering for håndtering av viktige prosessdata ..... 18
2.3 AVSLUTTENDE KOMMENTARER ..... 18

**3 VERKTØY FOR VISUALISERING AV STORE OG KOMPLEKSE DATAMENGDER ..... 19**

3.1 GENERELLE KRAV TIL VISUALISERINGSVERKTØY ..... 19
3.2 THE INFOVIS TOOLKIT ..... 19
3.3 XMDVTOOL ..... 20

**4 DESIGN AV PROTOTYP ..... 21**

4.1 BAKGRUNN ..... 21
4.2 UTVIKLING ..... 22
4.2.1 Brukergrensesnitt ..... 23
4.2.2 Koblinger mellom prosess-, produkt- og produktnivåbildet ..... 24
4.2.3 Informasjonsvisualisering ..... 25
4.3 DEMONSTRASJON OG EVALUERING ..... 28

**5 CASE STUDY ..... 30**

5.1 BAKGRUNN ..... 30
5.2 PRODUKSJONSBESKRIVELSE ..... 32
5.3 PROTOTYP ..... 33
5.4 DEMONSTRASJON OG EVALUERING ..... 34
5.4.1 Deltakere ..... 34
5.4.2 Gjennomføring av demonstrasjonen ..... 34
5.4.3 Resultater ..... 34

**6 KONKLUSJON OG FREMTIDIGE UTFORDRINGER ..... 36**

6.1 KONKLUSJON ..... 36
6.2 VIDERE ARBEID OG FREMTIDIGE UTFORDRINGER ..... 37

**7 REFERANSER ..... 38**

**VEDLEGG A: RÅDATA OG UTREGNINGER ..... 43**

**VEDLEGG B: KILDEKODE ..... 48**

# Figurliste

FIGUR 1 - TRESTRUKTUR [JON98] ..... 7
FIGUR 2 - TREEMAPS [JON98] ..... 9
FIGUR 3 - CAM TREE ($90^O$ ROTERING AV ET CONE TREE) [RMC91] ..... 10
FIGUR 4 - RINGS: A TECHNIQUE FOR VISUALIZING LARGE HIERARCHIES [TM02] ..... 11
FIGUR 5 - KARTESISKE KOORDINATER [JON98] ..... 12
FIGUR 6 - KOORDINATENE PRESENTERT VED HJELP AV PARALLELLKOORDINATER [JON98] ..... 12
FIGUR 7 - PARALLELLKOORDINATER I 4 DIMENSJONER [JON98] ..... 13
FIGUR 8 - EKSEMPLER PÅ GLYPHSTYPER [WAR02] ..... 14
FIGUR 9 - TABELL-LINSE (TABLE LENS) [RC94], (KILDE: WWW.INXIGHT.COM) ..... 16
FIGUR 10 - TRADISJONELLE STOLPEDIAGRAMMER [KH+02] ..... 17
FIGUR 11 - PIKSELBASERTE STOLPEDIAGRAMMER [KH+02] ..... 17
FIGUR 12 - PARALLELLKOORDINATER AV PRODUKSJONSDATA GENERERT AV XMDVTOOL ..... 20
FIGUR 13 - PROTOTYP ..... 23
FIGUR 14 - KOBLINGER MELLOM PROSESS-, PRODUKT- OG PRODUKTNIVÅBILDET ..... 24
FIGUR 15 - PARALLELLKOORDINATER ..... 25
FIGUR 16 - FORBEDRET DESIGN AV PARALLELLKOORDINATER ..... 25
FIGUR 17 - STJERNEGLYPHS [SF+72] ..... 26
FIGUR 18 - VALG AV PRODUKT PÅ PRODUKTNIVÅ ..... 27
FIGUR 19 - VISUALISERING AV PARALLELLKOORDINATER VED VALGT PRODUKT ..... 27
FIGUR 20 - SPORINGSSTRUKTUREN, SCANWAFER III [FLE02] ..... 31
FIGUR 21 - PROTOTYP (SCANWAFER III) ..... 33

# 1 INTRODUKSJON

På grunn av økende krav til sporing og kvalitetskontroll av produksjonsprosesser, er behovet for gode kvalitetssikringssystemer blitt mer og mer aktuelt. Visualisering av slike prosesser vil kunne være gode hjelpemidler for effektivisering av kontroll-rutiner.

Hensikten med denne oppgaven er å foreta litteraturstudier for å søke etter visualiseringskonsepter som kan benyttes i produksjonsprosesser med store data-mengder og hvor det er viktig å kunne illustrere disse instruktivt på relativt liten plass. Oppgaven har ikke som hensikt å finne det mest optimale visualiserings-konseptet.

En produksjonsprosess generer data som ofte lagres i databaser med egne skjemaoppsett, datamodell og databasespråk [DC01]. Visualiseringsutfordringen vil være å finne konsepter som kan samle relevante data fra disse databasene og presentere produksjonsdataene på en oversiktlig og håndterbar måte sett i forhold til hensikten; sporing og kvalitetskontroll og muligheten for å treffe tiltak hvis avvik inntreffer.

De prosessene som skal undersøkes og visualiseres vil være produksjonsprosesser som benyttes i prosessindustrien. En slik produksjonssyklus kan for eksempel være å følge et råvareparti og se hvordan dette transformeres til mellomprodukter og produkter gjennom prosessen (se figur 20).

## 1.1 Emne og oppgavebeskrivelse

Emnet for denne oppgaven er presentasjon av produkt- og produksjonsdata for sporing av feil i produksjonen og kvalitetskontroll av sluttproduktet.

Oppgaven går ut på å kartlegge konsepter for informasjonsvisualisering av store og komplekse datamengder. Noen av disse konseptene skal presenteres i en prototyp som skal demonstreres og evalueres av en testgruppe ved ScanWafer III som er valgt som produksjonsprosess for å teste ut noen av konseptene. Fokuset vil dermed være informasjonsvisualisering av data generert i en produksjonsprosess. I tillegg skal prosesstopologi og trestrukturer studeres for å vise hvordan et produkt- og prosessbilde kan illustreres for å kunne gjennomføre en enkel traversering i strukturen.

## 1.2 Motivasjon

I dag stilles det strenge krav til kvalitetssikring samtidig som den stadig økende konkurransen har ført til større effektivitet i produksjonsprosessene. I tillegg til dette har flere og flere prosesser blitt automatisert. Det å ha god kontroll i produksjonsflyten og lav feilprosent på produserte produkter er derfor blitt en viktig og etterspurt del av overvåkingsrutinene. Systemer for god kvalitetssikring og sporing av feil kan gi bedrifter både økonomiske fordeler og konkurransefortrinn i forhold til konkurrentene.

Det er da viktig å ha gode systemer for presentasjon (visualisering) av innsamlede data som gir muligheter for å gjennomføre tiltak når avvik inntreffer. Slike presentasjoner er ofte basert på tradisjonelle måter å vise produksjonsprosessdata på som for eksempel tabellstrukturer og diagrammer basert på aggregerte data. Svakheter ved dette er at slike tabellstrukturer og diagrammer ofte krever stor plass og er lite oversiktlige. Det burde derfor ved valg av alternative datavisualiserings-konsepter, være potensialer for forbedringer som vil gi brukerne kraftigere verktøy for hurtig og effektiv tilgang til kritiske data.

## 1.3 Utfordringer og oppgavebegrensning

Da det i en produksjonsprosess genereres store mengder data, vil utfordringene være å presentere et lett oversiktelig produkt- og prosessbilde samtidig som brukeren får god oversikt over de viktige produksjonsdataene.

Informasjonsvisualisering er et stort område hvor det foregår mye forskning. Det er derfor nødvendig å begrense oppgaven til fortrinnsvis å gjelde produktdata-visualiseringen og legge mindre vekt på alarmer og andre data som oppstår gjennom en produksjonssyklus. Dette er også viktige data, men for at denne oppgaven ikke skal bli for omfattende vil disse dataene ikke bli vektlagt og vil derfor ikke analyseres. Dette kan være et emne for senere analyser av produksjonsprosesser.

## 1.4 Metode og mål

Metodene som er valgt for denne oppgaven er å gjennomføre et litteraturstudie for å kartlegge konsepter for visualisering av trestrukturer, prosesstopologi og store og komplekse datamengder og samtidig beskrive hvor langt forskningen har kommet innenfor de gitte emnene. På bakgrunn av dette studiet skal en produksjonsprosess studeres, og det skal bygges en prototyp ut fra den prosessen som er valgt som case study og de idéene som har kommet frem i løpet av litteraturstudiet. Til slutt skal denne prototypen demonstreres og evalueres.

Målet med oppgaven er å illustrere en effektiv måte å vise store datamengder på slik at brukerne av systemet lett kan få tilgang på de dataene som de kan benytte til å spore feil og kvalitetsforringelser av de produktene som blir produsert i produksjonssyklusen.

## 1.5 Organisering av rapporten

Rapporten er organisert slik at det først gis en oversikt over konsepter og verktøy for visualisering av trestrukturer, prosesstopologier og store og komplekse datamengder. Disse relateres til artikler innenfor de gitte emnene for å kartlegge forskningsstatus. Deretter skal prototypens utviklingen og funksjonalitet beskrives.

I kapittel 5 beskrives et case study som ble gjennomført for å underbygge noen av de valgte konsepter som ble gjort i designet av prototypen og for å gi en tilbakemelding på om disse konseptene kunne la seg benytte i en produksjons-prosess.

Til slutt kommer hovedkonklusjonen og en diskusjon rundt fremtidige utfordringer for videreutvikling av de konseptene som er blitt gjennomgått, og eventuelt nye idéer som kan være interessante å vurdere ved en mulig senere videreutvikling av prototypen.

# 2 KONSEPTER FOR PRESENTASJON AV STORE OG KOMPLEKSE DATAMENGDER, TRESTRUKTURER OG PROSESSTOPOLOGIER

Denne delen skal gi en innføring i konsepter som eksisterer for visualisering av trestrukturer, prosesstopologier og store og komplekse datamengder, samtidig som leseren skal få en oversikt over hvor forskningen står innenfor disse emnene.

## 2.1 Bakgrunn

Etter at datamaskiner og annet lagringsutstyr ble billig, åpnet virkelig mulighetene seg for å visualisere store datamengder. De siste tre tiårene med datavisualisering og særlig innenfor visualiseringsutvikling av multidimensjonale, multivariable data, kan røft deles inn i fire perioder. Disse periodene er *søkeperioden* før 1977, *oppvåknings-perioden* mellom 1977 og 1985, *oppdagelsesperioden* mellom 1986 og 1991 og *vurderings- og utviklingsperioden* fra 1992 og frem til i dag. Forskere har studert multivariable data siden 1782 da Crome brukte punktsymboler for å vise geografisk distribusjon i Europa av 56 handelsvarer.

1977 ble et veiskille ved utgivelsen av John W. Tukeys bok «Exploratory Data Analysis» [Tuk77]. Der forklarte han at utforsking av data er mer enn et verktøy, det er også en måte å tenke på. Det lærer mennesker hvordan de skal tolke informasjon fra data. Da personlige datamaskiner ble vanlige, ble dette forskernes kraftigste verktøy noensinne. Dette gjorde det mulig å visualisere data interaktivt i flere enn to dimensjoner. De lange og vanskelige kalkulasjonene ble plutselig tilgjengelige i sanntid. Fra 1977 til 1985 var to- og tredimensjonale data de mest vanlige i forskningssammenheng, men på slutten av perioden ble mer og mer oppmerksomhet rettet mot multivariable, spatiale data. Et eksempel på dette er Daniel Asimovs «Grand Tour Technique» for å undersøke multivariable data som ble projisert til et todimensjonalt plan [Asi85].

I 1987 erklærte «NSF (National Science Foundation) workshop» formelt behovet for to- og tredimensjonal spatial objektvisualisering. Dette førte til at forskere satte av mye tid til å utvikle presentasjonskonsepter for visualisering av multi-dimensjonale, multivariable data. Problemene i de tidligere periodene med tilgjengeligheten av høyhastighetsgrafikkmaskinvare var på mange områder løst. Forskningen ble derfor rettet vekk fra utforsking av data til fargefull, flere-dimensjonal grafikk. Denne perioden sluttet med «IEEEs Visualization '91 Conference». Etter denne konferansen har mye av forskningen vært fokusert mot vurdering og utvikling av konsepter som ble beskrevet i tidligere perioder.

Denne bakgrunnshistorien er basert på en artikkel som beskriver en undersøkelse av tre tiår med multidimensjonale, multivariable visualiseringsteknikker. [WB97]

## 2.2 Visualisering

Visualisering av data har en lang historie [WB97], og det finnes store mengder med artikler innenfor dette feltet. I denne delen vil noen artikler som tar for seg generelle problemer med visualiseringsteknikker i forhold til produksjonsprosesser og store datamengder bli beskrevet for så å gå mer i detalj på noen spesifikke konsepter som er vurdert under arbeidet med denne oppgaven.

Eksempler her er konsepter innenfor visualisering av trestrukturer og prosess-topologier og visualisering av store og komplekse datamengder. En produksjons-prosess beskrives som en prosesstopologi som er en trestruktur med en flatere toppstruktur enn tradisjonelle trestrukturer. Det er derfor viktig å se på konsepter som kan optimalisere visualiseringen av trestrukturer. Da datamengden i en produksjonsprosess kan bli stor og kompleks og derfor vanskelig å visualisere ved hjelp av tradisjonelle visualiseringskonsepter, er det nødvendig å studere spesifikke konsepter som er designet med tanke på å visualisere store datamengder på begrensede flater.

### 2.2.1 Generelle problemer med visualiseringsteknikker

Interaktive teknikker er i følge Mei C. Chuah og Steven R. Roth, kraftige verktøy for å manipulere visualiseringer med den hensikt å analysere, kommunisere og hente informasjon. Dette gjelder spesielt for 3D-visualiseringer. Selv om det har blitt introdusert mange nye måter for interaktiv navigering i den senere tid, har det blitt gjort lite for å forklare hva komponentene kan brukes til, hvordan forskjellige teknikker er relatert til hverandre og hvordan de kan kombineres for å gjøre visualiseringsverktøyet mer fleksibelt. Forfatterne har bygd opp et rammeverk som skal løse slike problemer ved å lage et system for basisvisualiseringsinteraksjon (BVI). [CR96]

Visage er et brukergrensesnittmiljø for utforsking av informasjon. Forskerne bak denne artikkelen har også her sett det som et problem at visualiseringsteknikker bare er utarbeidet for det formålet det er tenkt brukt for å løse og ikke på et generelt grunnlag slik at det kan benyttes i andre sammenhenger. De har derfor laget en prototyp som benytter seg av flere visualiseringsteknikker for å gjøre utforskingen av informasjonen bedre ved å la brukeren få mulighet til å visualisere den samme informasjonen på flere måter. [RL+96]

Alf Ove Braseth m. fl. har sett på nye metoder for å designe skjermkomponenter til bruk i kontrollrom innenfor petroleumssektoren. Da det er begrenset hvor store skjermene som brukes til overvåking er, har de brutt med gamle tradisjoner om å illustrere komponenten slik de ser ut i produksjonen og gitt eksempler på hvordan flere komponenter kan plasseres innenfor det gitte området ved å bygge disse opp på en enklere måte. Dette gjør at operatørene får bedre helhetlig oversikt over produksjonen. Et annet viktig element som de også tilfører komponentene er å vise hvordan for eksempel temperaturen har utviklet seg over tid for å gjøre det enklere for operatørene å se stabiliteten til de forskjellige komponentene. [BWV03]

Under visualiseringsdesign er det også viktig å ta hensyn til bruk av farge-kombinasjoner. Ved å benytte prinsipper for lagvisualisering i kartografi kan dette danne grunnlag for perseptuelt hierarkisk informasjonsvisualisering. Ti prinsipper for slik lagdeling er utviklet, og metoden for visuell lagdeling basert på fargekoding er illustrert basert på et typisk kontrollromsskjermbilde. [vLa01]

Det er også gjort en brukerundersøkelse basert på dette studiet. Denne undersøkelsen viste at lagdeling basert på fargekoding kan gi en signifikant raskere responstid enn gråskala- eller ikke-lagdelte metoder. [vLD02].

### 2.2.2 Visualisering av trestrukturer og prosesstopologier

Utgangspunktet for visualisering av trestrukturer og prosesstopologier er ganske like. Det som skiller dem er at prosesstopologier typisk har en mye flatere topp-struktur enn tradisjonelle trestrukturer. Prosesstopologier har som regel ikke bare én rot, men flere røtter på toppnivået. Et tre er en spesialisering av en graf og kan defineres som en koblet graf som ikke er satt sammen i en krets [GT87].

Det er forsket mye på å lage algoritmer for hvordan tradisjonelle trestrukturer eller grafer kan visualiseres [RT81, MHM00a, MHM00b, NH03]. Det har også blitt gjennomført kognitive målinger av estetiske visualiseringer av grafer [WP+02].

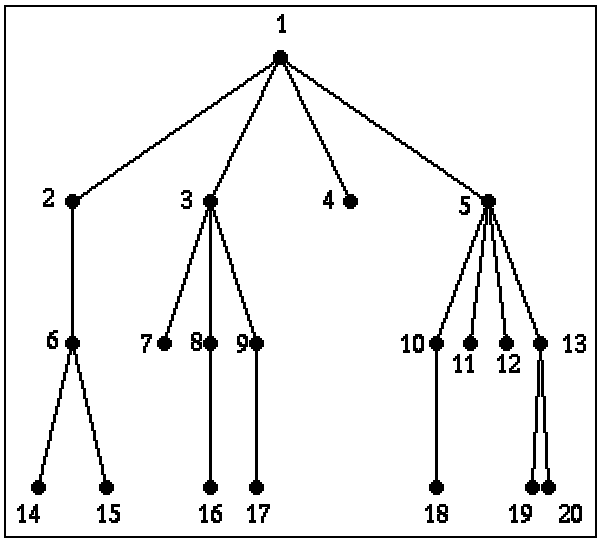
I løpet av de to siste tiårene har også andre måter å visualisere slike strukturer på blitt presentert. Eksempler på dette er treemaps [Shn92, TJ92], 3D-løsninger [RMC91, HK97] og ringløsninger [AH98, MH98, TM02]. En evaluering av tre 3D-modeller er også gjennomført [WCJ98].

Metodene og resultater fra flere av disse artiklene vil bli beskrevet i gjennomgangen av de respektive konseptene.

## Trestruktur

Standard terminologi for trestrukturer er at de har én rot for treet. Denne roten har ingen foreldre, men den har subtrær som kan beskrives som barn av denne roten. Foruten denne hovedroten har hvert nivå røtter som har både fedre og barn. Fedrene er røttene i nivået over. [Knu97]

Figur 1 viser et eksempel på en slik trestruktur. Denne strukturen har en lineær orientering.




**Figur 1 - Trestruktur [Jon98]**

Edward M. Reingold og John S. Tilford har studert forskjellige algoritmer som er blitt lagt frem for å produsere ryddige og ordnete visualiseringer av trestrukturer. Dette er grafvisualiseringer som er estetisk behaglig å se på og som benytter minimalt med plass. Basert på dette studiet presenterer de en ny algoritme med ett estetisk tilleggskriterium i forhold til tradisjonelle kriterier. De tradisjonelle kriteriene som er illustrert i figur 1, er at noder på samme nivå skal ligge langs en rett linje og at de rette linjene som definerer nivåene skal være parallelle, et venstrebarn skal plasseres til venstre og et høyrebarn skal plasseres til høyre og det siste kriteriet er at farsnoden skal sentreres over sønnene sine. Tilleggskriteriet de tilfører tar utgangspunkt i at det er tilfredstillende at symmetriske trestrukturer blir visualisert symmetrisk og å definere at et tre og dets speilbilde skal produsere visualiseringer som reflekterer hverandre. Konklusjonen er at en ulempe ved dette kriteriet er at bredden har en tendens til å bli større enn strengt tatt nødvendig, men grafvisualiseringen blir generelt «behagelig» å se på. [RT81]

M. Scott Marshall m. fl. bygger videre på idéene til Reingold og Tilford [RT81], og ser på mulighetene for automatisk generering av interaktive oversiktsdiagram for navigering av store grafer. Denne løsningen er ikke bare begrenset til å illustrere

grafer, men kan også visualisere en trestruktur av et hvert datasett på grunn av dens hierarkiske struktur. [MHM00b]

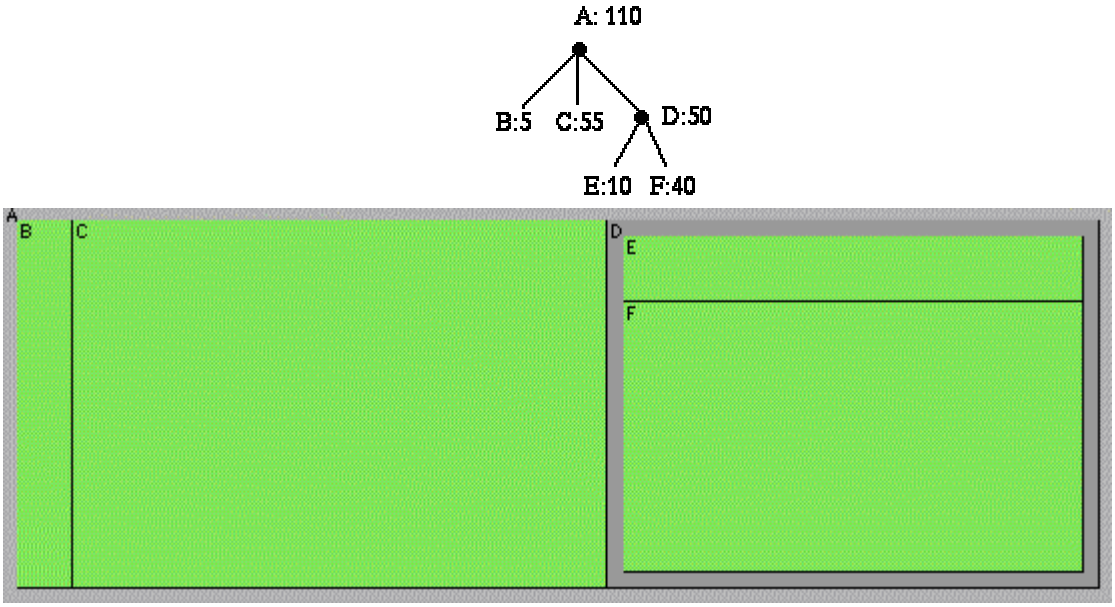
M. Scott Marshall m. fl. har også laget et rammeverk basert på objektorientert design for grafvisualisering. Dette rammeverket består av et bibliotek av programmeringsklasser for blant annet programmeringsspråket Java som kan benyttes for å visualisere nettverk- og trestrukturer. [MHM00a]

Colin Ware m. fl. har gjennomført kognitive målinger av estetiske visualiseringer av grafer. Basert på et kriterium om den korteste veien mellom noder i grafen har de sett på forskjellige faktorer som for eksempel kontinuitet, antall krysninger av linjer og lengden av korteste vei. Med bakgrunn i disse faktorene har de gjennomført en brukerundersøkelse med 43 deltakere. Undersøkelsen ble gjennomført ved at deltakerne først fikk dokumentasjon om grafvisualisering for å gjøre seg kjent med område. Deretter ble testgrafer vist før selve grafvisualiseringen ble gjennomført. Under testen ble data om responstid og nøyaktighet hos deltakerne i forhold til de eksperimentelle diagrammene samlet inn. Forfatterne mener at det største bidraget denne undersøkelsen ga var selve metodikken for gjennomføringen av bruker-testene. [WP+02]

## Treemaps

Treemaps [Shn92] har sin opprinnelse fra venndiagram. De er designet for en spesiell klasse av trestrukturer kalt katalogtrestrukturer. En nummerisk verdi er assosiert med hver node i katalogtreet som beskriver størrelsen av filene som er i subtreet som hører til noden. Hver node er visualisert som et rektangel med en størrelse proporsjonal til dets verdi. Alle etterkommerne av noden er vist som rektangler inne i dets rektangel. Treemaps er en effektiv metode for å vise store trær på begrenset skjermplass. [Jon98]

Figur 2 viser hvordan et enkelt tre kan visualiseres ved hjelp av treemaps.



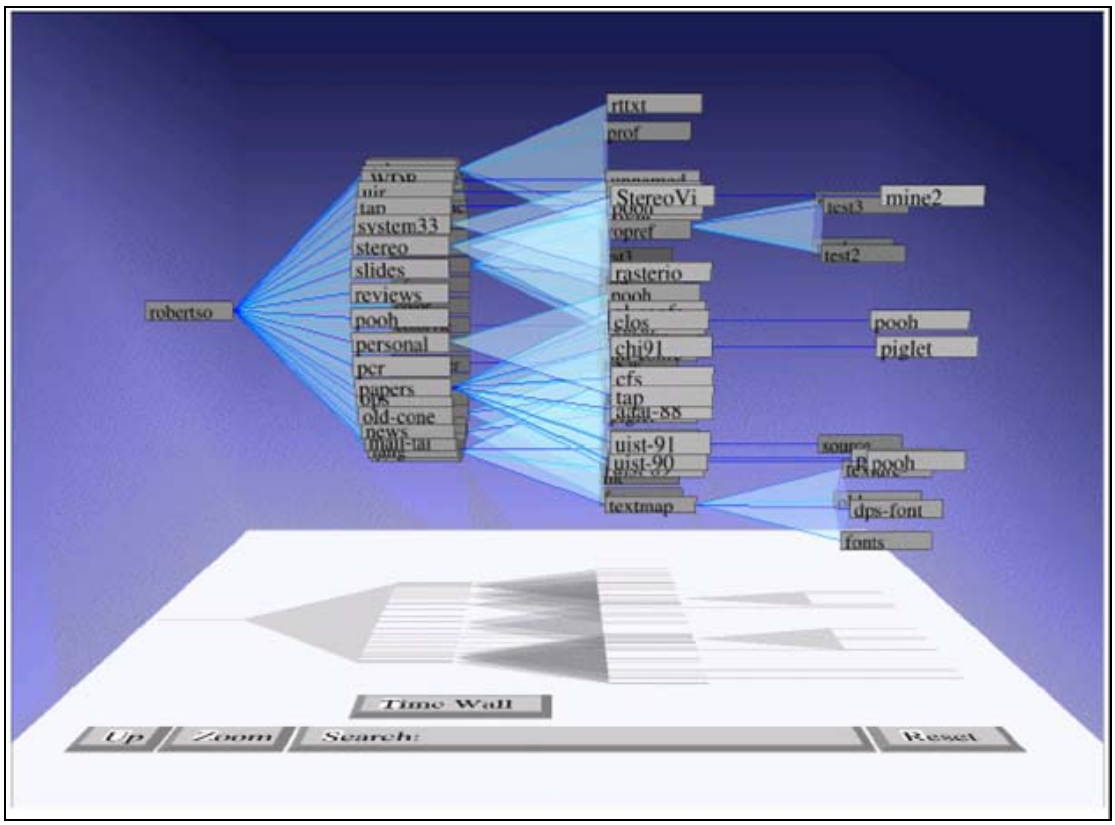

**Figur 2 - Treemaps [Jon98]**

Treemaps representerer en unik mulighet for effektivt å kommunisere informasjon om store hierarkier som inneholder informasjon som brukere har samlet over en lengre tidsperiode eller helt oversett. Ved å gi brukerne en mulighet til å vise hele hierarkier gjør dette det mulig og hente informasjon som kanskje ikke har vært opplagt via tradisjonelle metoder. Mulige applikasjoner hvor treemaps kan benyttes vil være hierarkier, salgsdata, aksjeporteføljeanalyser, organisasjonskart og medisinske data. [TJ92]

## 3D-løsning

Oppgaven med å håndtere og aksessere store informasjonsflater er et problem. Nye teknologier for 3D-visualisering og interaktiv animasjon kan tilby mulige løsninger på dette problemet, spesielt når strukturen for informasjonen kan bli visualisert. George G. Robertson m. fl. har laget en hierarkisk 3D-visualisering de har kalt Cone Tree, det vil si et kjegleformet tre. Toppnoden er plassert øverst og dets barn er jevnt fordelt som en kjegle under hvor barna igjen er plassert kjegle-formet under disse igjen. Når en node blir valgt roteres treet slik at denne noden kommer i front og blir uthevet. [RMC91]

Figur 3 gir et eksempel på en slik visualisering som er kalt Cam Tree. Dette er en 90$^{o}$ rotering av Cone Tree for å lage en projisert todimensjonal skygge slik at den visualiserte informasjon om den hierarkiske strukturoppbyggingen kommer bedre frem. [RMC91]




**Figur 3 - Cam Tree (90$^{o}$ rotering av et Cone Tree) [RMC91]**

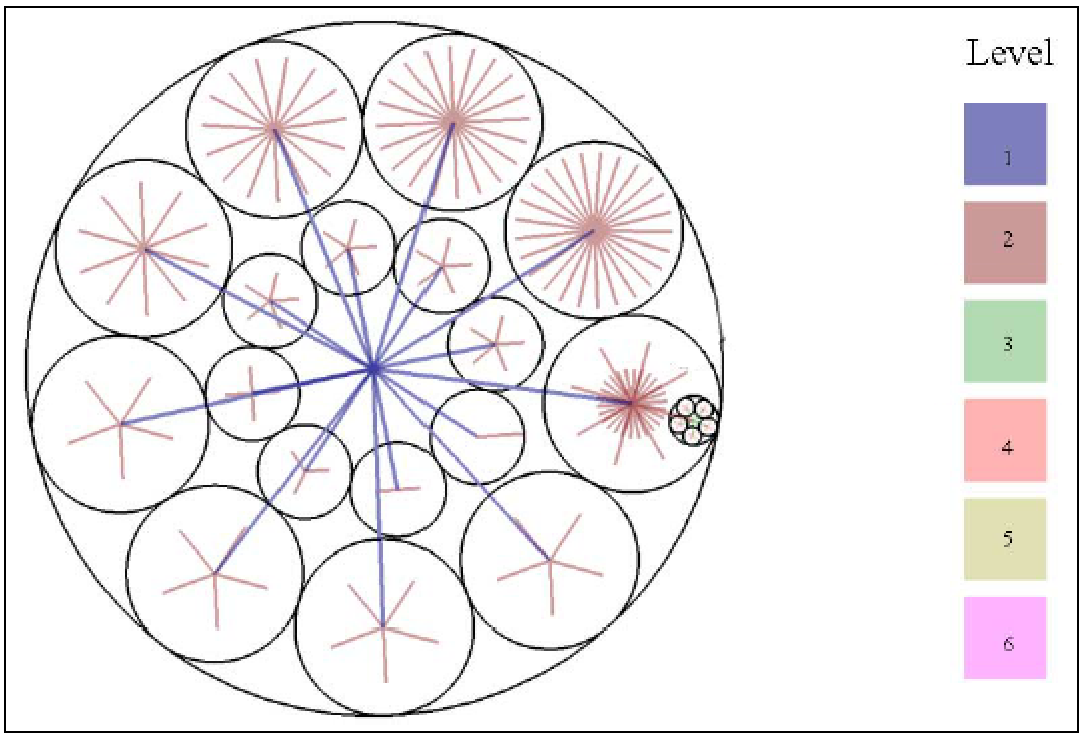
Ulrika Wiss m. fl. har foretatt en evaluering av tre 3D-modeller. Disse modellene er «Cam Tree», «Information Cube» og «Information Landscape». Konklusjonene var at disse modellene fungerte bra for det problemet de i utgangspunktet var designet for, men ved å benytte andre datasett kunne de fort vise forskjellige resultater og slike problemer var sjeldent nevnt i beskrivelsen av systemet. Ved valg av visualiseringsmodell er det viktig at slike problemer er nevnt, og håndtering av slike problemer gir også mulighet for gjenbruk av gamle visualiseringsmodeller. [WCJ98]

## Ringløsning

Dette er en forholdsvis ny metode for å presentere store trestrukturer eller grafer. Denne metoden går ut på å spesifisere områder for primær og sekundær fokus og også gi mulighet for å vise multippel fokus uten å miste forståelsen for opp-byggingen av grafen. Styrken ved denne metoden er dens evne til å vise et større

område i fokus og også gi mer utfyllende informasjon i dette område enn visualisering av tradisjonelle grafer. [TM02]

Figur 4 gir et eksempel på en slik ringløsning. Dette er et forsøk på å optimalisere tidligere ringvisualiseringer utviklet blant annet av Guy Melançon og Ivan Herman [MH98]. Deres idéer bygger igjen videre på kompleksiteten og metodikken som ligger til grunn for Reingold og Tilfords algoritme for grafvisualiseringer [RT81].




**Figur 4 - RINGS: A Technique for Visualizing Large Hierarchies [TM02]**

Keith Andrews og Helmut Heidegger presenterer en teknikk for visualisering og utforsking av store hierarkier basert på trinnvise, halvsirkulære flater kalt informasjonsstykker. Denne teknikken går ut på at hierarkiene blir visualisert ved hjelp av en eller flere halvsirkulære flater og hver flate representerer flere nivåer av et hierarki. Dette arbeidet er fortsatt under utviklingen slik at forfatterne ikke har kommet med noen konklusjoner ennå. [AH98]

### 2.2.3 Visualiseringsteknikker for store og komplekse datamengder

Forskningen innenfor informasjonsvisualisering og visualiseringsteknikker for håndtering av store og komplekse datamengder blir stadig mer omfattende. Eksempler på visualiseringskonsepter er parallellkoordinater [ID90, Ins99], glyphs [SF+72, Che73, Bed90, KE01, War02], tabellstrukturer [RC94, SBB96] og pikselbaserte stolpediagrammer [KH+02, KP+03].

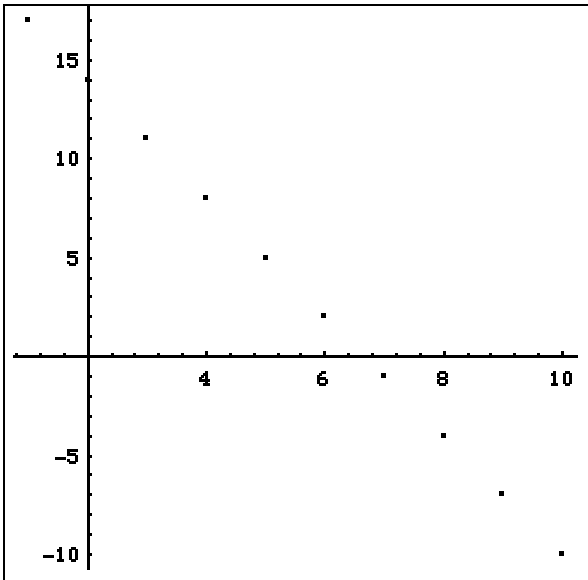
Metodene og resultater fra flere av disse artiklene vil bli beskrevet i gjennomgangen av de respektive konseptene.

**Parallellkoordinater**

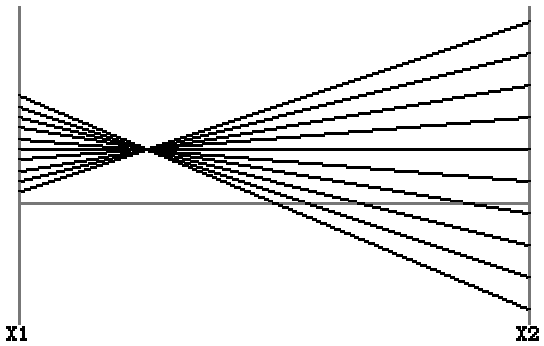
Parallellkoordinater er en teknikk for å presentere multidimensjonale data og ble introdusert av Alfred Inselberg på slutten av 70-tallet [Ins99] og senere i samarbeid med Bernard Dimsdale [ID90]. Resultatene fra dette samarbeidet ble presentert på IEEEs første visualiseringskonferanse i 1990 og førte til at dette visualiserings-konseptet også ble utforsket videre av andre forskere [FWR99, GK03].

I tradisjonelle kartesiske koordinater er alle aksene innbyrdes vinkelrette, mens for parallellkoordinater er aksene parallelle i forhold til hverandre og avstanden mellom aksene er like stor [Jon98]. Et punkt i kartesiske koordinater representerer en linje i parallellkoordinater.

I figur 5 er en serie av punkter tegnet langs linja, $x_2=-3x_1+20$, i kartesiske koordinater. Figur 6 viser en presentasjon av de samme punktene i parallellkoordinater.

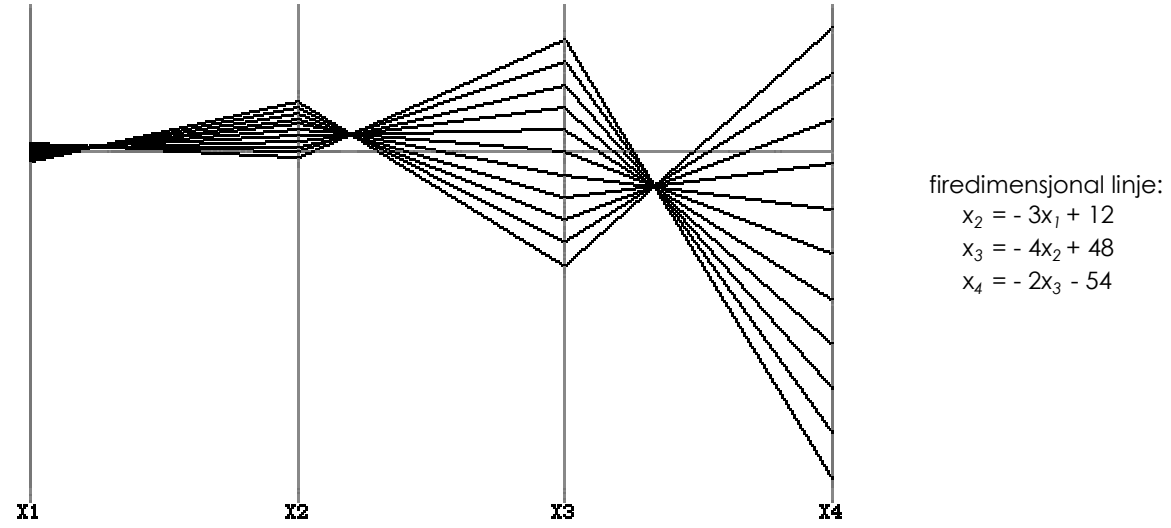



**Figur 5 - Kartesiske koordinater [Jon98]**




**Figur 6 - Koordinatene presentert ved hjelp av parallellkoordinater [Jon98]**

Figur 7 viser parallellkoordinatpresentasjon av en serie med punkter langs en firedimensjonal linje.

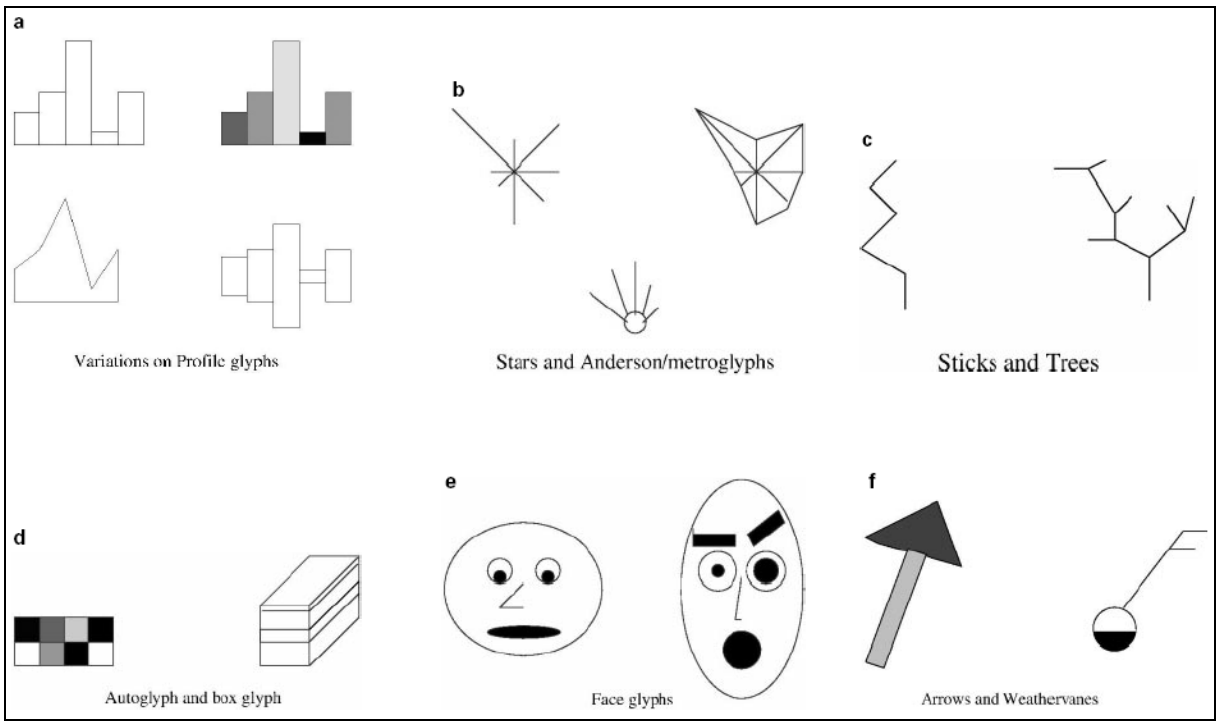



**Figur 7 - Parallellkoordinater i 4 dimensjoner [Jon98]**

## Glyphs

Glyphs er grafiske entiteter som består av en eller flere dataverdier basert på attributter som for eksempel form, størrelse, farge og posisjon. De har blitt mye brukt innenfor visualisering av data og informasjon, og de er spesielt egnet for å vise komplekse, multivariable sett av data. Plasseringen eller glyphsutformingen på en skjerm kan kommunisere signifikant informasjon i forhold dataverdier selv så vel som relasjonene mellom datapunktene. [War02]

Figur 8 gir eksempler på glyphs som har blitt brukt i forskjellige sammenhenger.




**Figur 8 - Eksempler på glyphstyper [War02]**

Den kanskje mest kjente av disse er ansiktene (face glyphs, figur 8e) som ble benyttet av Herman Chernoff for å visualisere statistiske fossile og geologiske data. Formålet var å presentere de multivariable dataene på en slik måte at de som skulle studere disse raskt kunne tilegne seg relevant informasjon, og basert på denne informasjonen velge passende statistiske analyseverktøy. [Che73]

Herman Chernoff og M. Hazeeb Rizvi har også foretatt en brukertest av Chernoffs glyphs basert på ansikter. Deler av konklusjonen på denne testen gikk ut på at under visse forhold gir ansiktuttrykkene relativt lite informasjon. For eksempel hvis øynene er små, vil posisjonen til pupillene bli borte og dermed vil viktig informasjon gå tapt, men i flere tilfeller gav disse glyphspresentasjonene tilfredstillende resultater. [CR75]

Jeff Beddow benytter autoglyphs (figur 8d) for å visualisere store og komplekse datasett på små dataskjermer. Datasettet bestod av omtrent 35 parametere og deres utvikling gjennom en 20-års periode, men Beddow benyttet bare 13 av disse. Hovedformålet var å se på relasjonene mellom solarvinder og jordas magnetiske felt. Gjennom utvikling av en prototyp og gjennomføring av et eksperiment som gikk ut på å avbilde alle parameterne simultant for å se om det dukket opp noen

globale eller lokale mønstre, konkluderte han med at visualisering ved hjelp av slike glyphs kan presentere mer komplekse datasett en standardmetoder tillater. [Bed90]

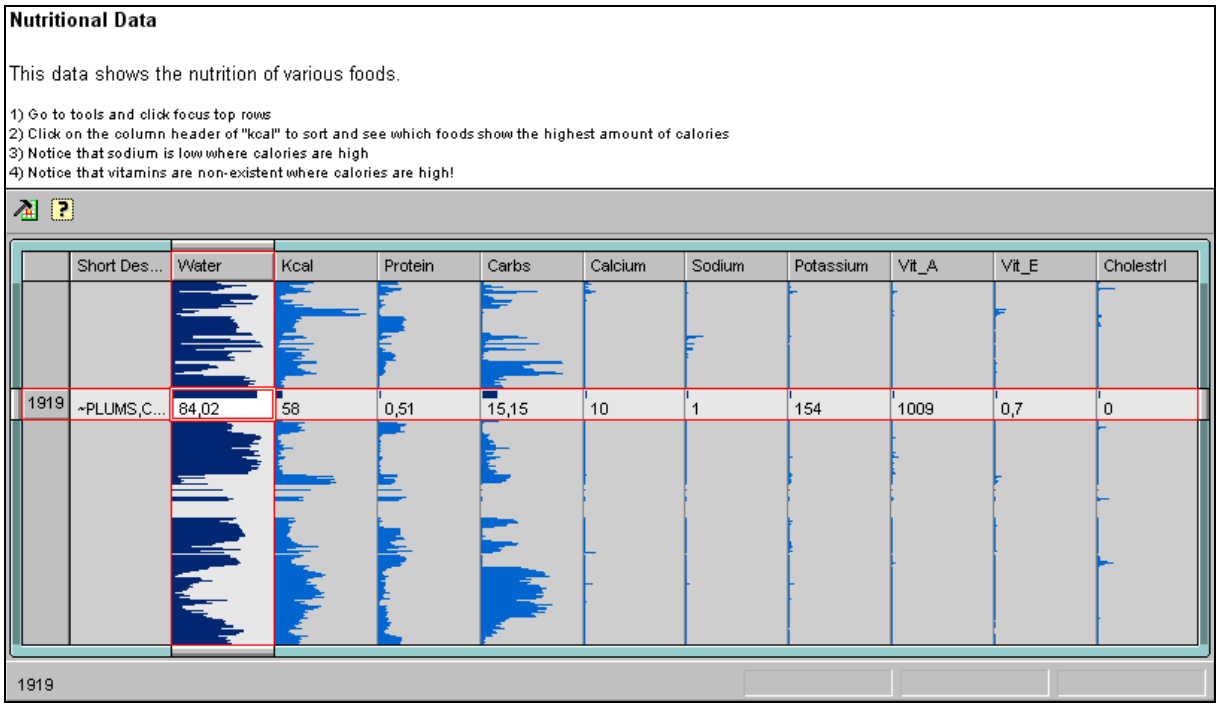
John H. Siegel m. fl. visualiserer medisinske data ved hjelp av stjerneglyphs (star glyphs, figur 8b) for å gjenkjenne forskjellige mønstre. De konkluderer med at å visualisere dataene på denne måten gir et mer troverdig bilde av resultatene enn antagelser de har opparbeidet ved å studere parameterne hver for seg. [SF+72]

Martin Kraus og Thomas Ertl har designet et system som gjør det mulig for ikke-programmerere å visualisere, utforske og analysere multivarable data ved å designe sine egne glyphspresentasjoner ved hjelp av minimal brukerinteraksjon. Dette systemet er testet ut på bilproduksjonsdata fra BMW-gruppen og forfatterne er overbevist om at slike visualiseringsteknikker vil gi store fordeler ved analysering av produksjonsdata. [KE01]

**Tabellstruktur**

Denne metoden går ut på å visualisere store datamengder som tabeller og lage en fokus + innhold (fisheye [Fur99]) teknikk slik at viktig informasjon kan bli vist hurtig. Motivasjonen bak denne tabellstrukturteknikken er den naturlige oppbyggingen av tabeller. Det mest fremtredende kjennetegnet til en tabell er tabellens lovbundethet i forhold til innholdet; informasjonen langs radene eller kolonnene er internrelatert og kan i visse sammenhenger leses som et koherent sett (for eksempel medlemmer av en gruppe eller atributter til et objekt). Tabell-linsestrukturen gjør interaksjon med mye større tabeller enn tradisjonelle tabeller enklere og kan komfortabelt vise 30 ganger flere celler enn de tradisjonelle tabellene. [RC94].

Figur 9 gir et eksempel på en slik tabellstruktur. Tabellstrukturen er visualisert slik at radene beskriver forholdet mellom parameterne for hvert produkt, mens kolonnene beskriver forandringen i parameterne ved valg av forskjellige produkter. Ved valg av et spesifikt produkt blir dette produktet uthevet samtidig som de tallmessige verdiene for dette produktet vises.



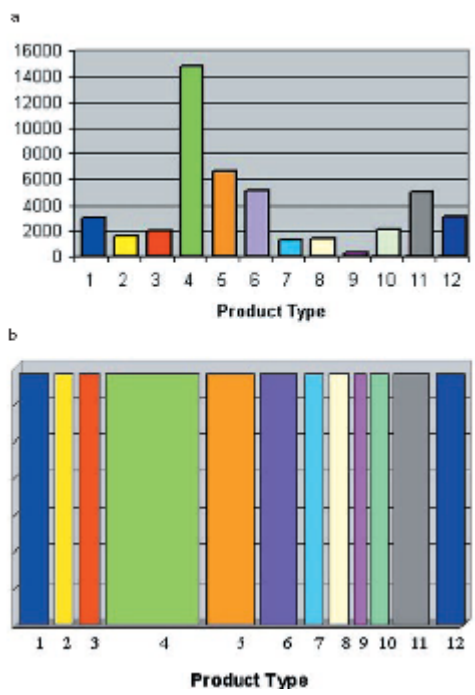

**Figur 9 - Tabell-linse (Table Lens) [RC94], (kilde: www.inxight.com)**

Michael Spenke m. fl. har også laget en tabell-linseløsning for datautforsking og interaktiv formulering av spørringer mot en database. Denne løsningen inkluderer den ovennevnte funksjonaliteten og i tillegg muligheten for dynamiske database-spørringer og dataaggregeringsmanipuleringer. [SBB96]

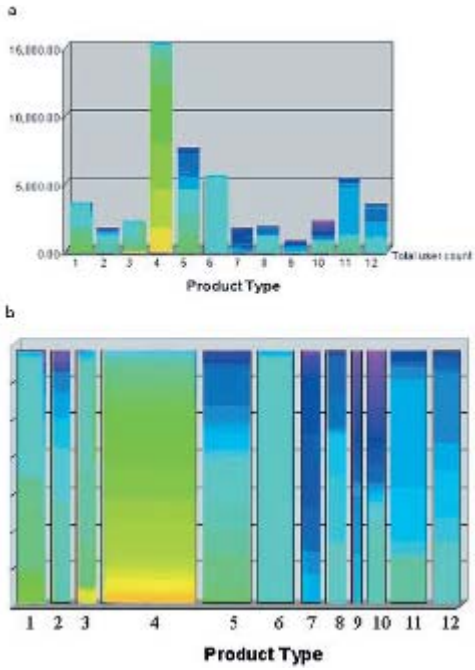
## Pikselbaserte stolpediagrammer

Enkel grafisk presentasjon er intuitivt og lett anvendelig, men dette viser som regel bare et lite utvalg av dataverdier som for eksempel ved bruk av tradisjonelle stolpediagrammer, og kan også ha en høy faktor av overlappende verdier som ødelegger for et signifikant antall av dataverdier (xy-plotting av verdier). Ved å benytte pikselrommet innenfor stolpene kan dette gjøre det mulig å visualisere store mengder av data. Dette gjør at de intuitivt ser ut som tradisjonelle stolpe-diagrammer, men i tillegg kan store mengder av detaljert informasjon bli presentert. [KH+02]

Figur 11 viser hvordan de tradisjonelle stolpediagrammene i figur 10 som viser kunders e-handelsdata, kan presenteres ved hjelp av pikselbaserte stolpe-diagrammer.




**Figur 10 - Tradisjonelle stolpediagrammer [KH+02]**




**Figur 11 - Pikselbaserte stolpediagrammer [KH+02]**

Bruken av tradisjonelle stolpediagrammer krever generelt en høy grad av datainnsamling, men visualiserer bare et lite antall data (bare 12 verdier er vist i figur 10a). Ved visualisering av store multidimensjonale data viser disse diagrammene bare et begrenset utvalg av verdier og informasjon som for eksempel datadistribusjon av multiple attributter, lokale mønstre, korrelasjon, trender og annen detaljert informasjon som for eksempel kundeprofiler (alder, inntekt, lokasjon og så videre) vises ikke. Tilegnes hver piksel innfor hver stolpe for eksempel en farge som illustrert i figur 11, kan det gi rom for visualisering av store deler av denne tilleggsinformasjonen som tradisjonelle stolpediagrammer ikke får visualisert. [KH+02]

### 2.2.3 Visualisering for håndtering av viktige prosessdata

Flere av visualiseringsteknikkene kan presentere så store mengder data på et skjermbilde at det blir viktig å lage filtrerings- og fokuseringsmekanismer for å fremheve de viktigste dataene.

Ying-Huey Fua m. fl. har sett på problemer ved bruk av visualiseringsteknikker hvis datamengdene blir enormt store og komplekse og beskriver mulige løsninger med utgangspunkt i parallellkoordinater på disse problemene. Deres løsning på problemene er å designe hierarkiske parallellkoordinater for å presentere et multi-oppløselig bilde av dataene med dertil egnete verktøy for navigering og filtrering for å muliggjøre oppdagelser av trender og mønster i datamengden. [FWR99]

Ved store datamengder vil det også bli et problem å følge de tradisjonelle linjene mellom parameterne i parallellkoordinater [ID90]. Martin Graham og Jessie Kennedy har sett på forbedringsmuligheter for å få bukt med dette problemet. De har byttet ut de tradisjonelle linjene med estetiske kurver for å bedre muligheten til å følge parameterne til hver enkel enhet. For å bedre visualiseringen ytterligere har de i tillegg sett på muligheten for spreding av punkter langs aksene med noen diskrete verdier for å separere like parametere mellom de forskjellige enhetene. [GK03]

## 2.3 Avsluttende kommentarer

Kapittel 2 viser at det finnes mange forskjellige konsepter for informasjons-visualisering. De er også ganske forskjellige i måten de presenterer dataene på og det er derfor viktig å analysere datagrunnlaget nøye før informasjonsvisualiserings-konseptet, eller kombinasjoner av konsepter, velges.

# 3 VERKTØY FOR VISUALISERING AV STORE OG KOMPLEKSE DATAMENGDER

Det finnes i dag mange verktøy for visualisering av store mengder data. En god oversikt er satt sammen av John Goodall [Goo03]. Flere av visualiserings-verktøyene er bygd opp ved å utvikle egne designpakker i for eksempel Java. Fordi disse må inkluderes på hver datamaskin som den utviklede prototypen skal demonstreres på, fører dette til at det fort blir ganske kompliserte installeringer som skal til for å benytte seg av designpakkene. Det er to visualiseringsverktøy som er blitt studert, The InfoVis Toolkit [Fek03] og XmdvTool [War94].

## 3.1 Generelle krav til visualiseringsverktøy

Det er flere krav som bør stilles til visualiseringsverktøy. For det første bør de være enkle og intuitive å bruke, det vil si at for å benytte dem bør man ikke trenge å sette seg inn i mangfoldige sider med dokumentasjon da funksjonalitet bør være selvskreven. I tillegg bør det være mulig å manipulere mange av teknikkene som er implementert for å kunne tilpasse disse teknikkene til det datasettet som skal visualiseres. En annen mulighet som bør være tilgjengelig er å kompilere det ferdige resultatet til en kjørbar fil, slik at minst mulig tid brukes til å få systemet operativt der det skal brukes. Hvis derimot det visualiserte resultatet skal benyttes på en annen plattform enn den det opprinnelig var designet for, bør det også være mulig å eksportere kildekode slik at det visualiserte resultatet kan kompileres på denne plattformen. Til slutt bør det også være muligheter for å eksportere noen av de visualiserte resultatene på en slik måte at de kan importeres og benyttes i et annet verktøy.

## 3.2 The InfoVis Toolkit

Dette verktøyet består av pakker utviklet i Java slik at det blir enklere å designe visualiseringskonsepter som kan benyttes i en prototyp. Inkluderte visualiserings-konsepter er blant annet spredningsplotting av data [BC87], tidsserier [Cle93], treemaps [Shn92] og lenkede nodediagram for trestrukturer og grafer. Muligheten for å bruke fisheyelinser for alle visualiseringsmetodene er også implementert.

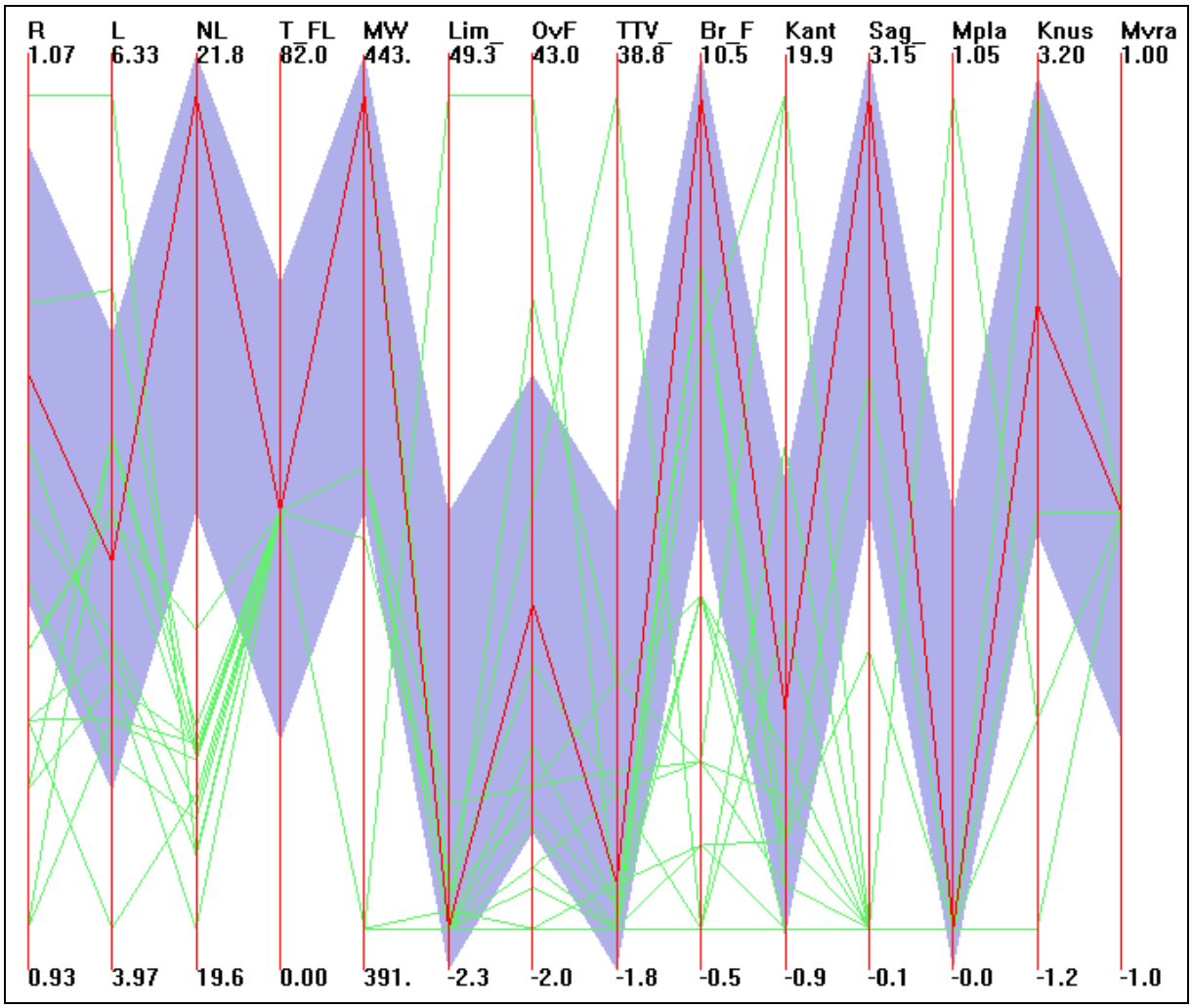
Problemet med dette verktøyet er at designpakkene, som er utviklet i Java, må inkluderes på hver datamaskin hvor prototypen skal demonstreres. Dette må gjøres fordi disse ikke er en del av standardpakkene som er inkludert i JRE (Java Runtime Environment) som er installert på de fleste datamaskiner i dag. Verktøyet ble derfor

ikke benyttet i prototypen, men under utviklingsprosessen ble noen av pakkene i verktøyet testet ut.

## 3.3 XmdvTool

Dette er en applikasjon hvor fire populære visualiseringsteknikker er implementert. Disse er spredningsplotting av data [BC87], dimensjonsstabling (hierarkisk visualisering) [LWW90], parallellkoordinater [ID90] og stjerneglyphs [SF+72]. Basert på datasett genererer denne applikasjonen visualiseringer basert på disse konseptene.

Dette verktøyet er blitt benyttet til å teste ut datasett fra produksjonsprosesser under utviklingen av prototypen. De stjerneglyphs som er benyttet i prototypen er generert ved hjelp av dette verktøyet. I figur 12 vises et eksempel på visualisering av parallellkoordinater generert av verktøyet.




**Figur 12 - Parallellkoordinater av produksjonsdata generert av XmdvTool**

# 4 DESIGN AV PROTOTYP

Dette kapittel skal beskrive utviklingen av prototypen. Før arbeidet med prototypen begynte, ble mulige visualiseringskonsepter som er nevnt i litteraturstudiet, diskutert med oppdragsgiver og representanter fra IFE. Demonstrasjoner og diskusjoner rundt utvikling av prototypen med de respektive partene er beskrevet i kapittel 4.3.

## 4.1 Bakgrunn

I dag benyttes en tradisjonell tabellstruktur med kolonner og rader for å vise data som er lagret i en database. Dataforespørsel skjer gjennom ferdigdefinerte rapporter. En slik løsning krever som regel ganske stor plass og det kan fort bli vanskelig å navigere i tabellstrukturen. Dette igjen kan føre til at viktige data blir oversett. I tillegg til denne løsningen er det også en mulighet for å vise produkt- og produksjonsdata i en enkel sporingsstruktur. Et problem med denne løsningen er at det er vanskelig å sammenligne produktenes parametere for å se forskjell på kvaliteten. En løsning som benyttes er å eksportere data til et regneark og prosessere dataene videre der.

I prototypen som er blitt designet for denne oppgaven, er det en videreutvikling av denne sporingsstrukturen av produkt- og produksjonsprosessen som danner grunnlaget. Den gamle løsningen beskrev denne sporingsstrukturen som en enkel tekstbasert struktur, mens i oppgavens prototyp er det valgt en mer visualisert modell for lettere å kunne traversere i strukturen. For å gjøre prosessen mer lesbar har produkt- og produksjonsprosessen blitt delt opp i hver sin struktur med koblinger mellom dem for å vise hvor man til en hver tid befinner seg i forhold til de to strukturene. Muligheten for å visualisere flere produkter på hvert produktnivå er det også tatt hensyn til ved å lage et produktnivåbilde. Ved å benytte en ring-struktur i dette bildet gis muligheten for å fokusere på et enkelt produkt og vise mer informasjon om dette. I tillegg til disse bildene blir produktenes parameter-verdier visualisert samtidig, noe som gjør det mulig å sammenligne produktenes kvalitet.

Den første prototypen var en forholdsvis enkel versjon utviklet i Microsoft PowerPoint som beskrev produkt- og prosessbildet og traverseringsmuligheter, som i sporingsterminologi kalles «drill-down», i denne strukturen. På dette stadiet var det ikke tatt hensyn til hvordan dataene som ble generert i produksjonen skulle visualiseres.

For å lage en mer fleksibel prototyp med videreutviklingsmuligheter, måtte et kraftigere verktøy tas i bruk. Etter å ha prøvd ut noen fritt distribuerte verktøy (se kapittel 3), ble avgjørelsen tatt om å designe et brukergrensesnitt i Java for å få mest mulig fleksibilitet.

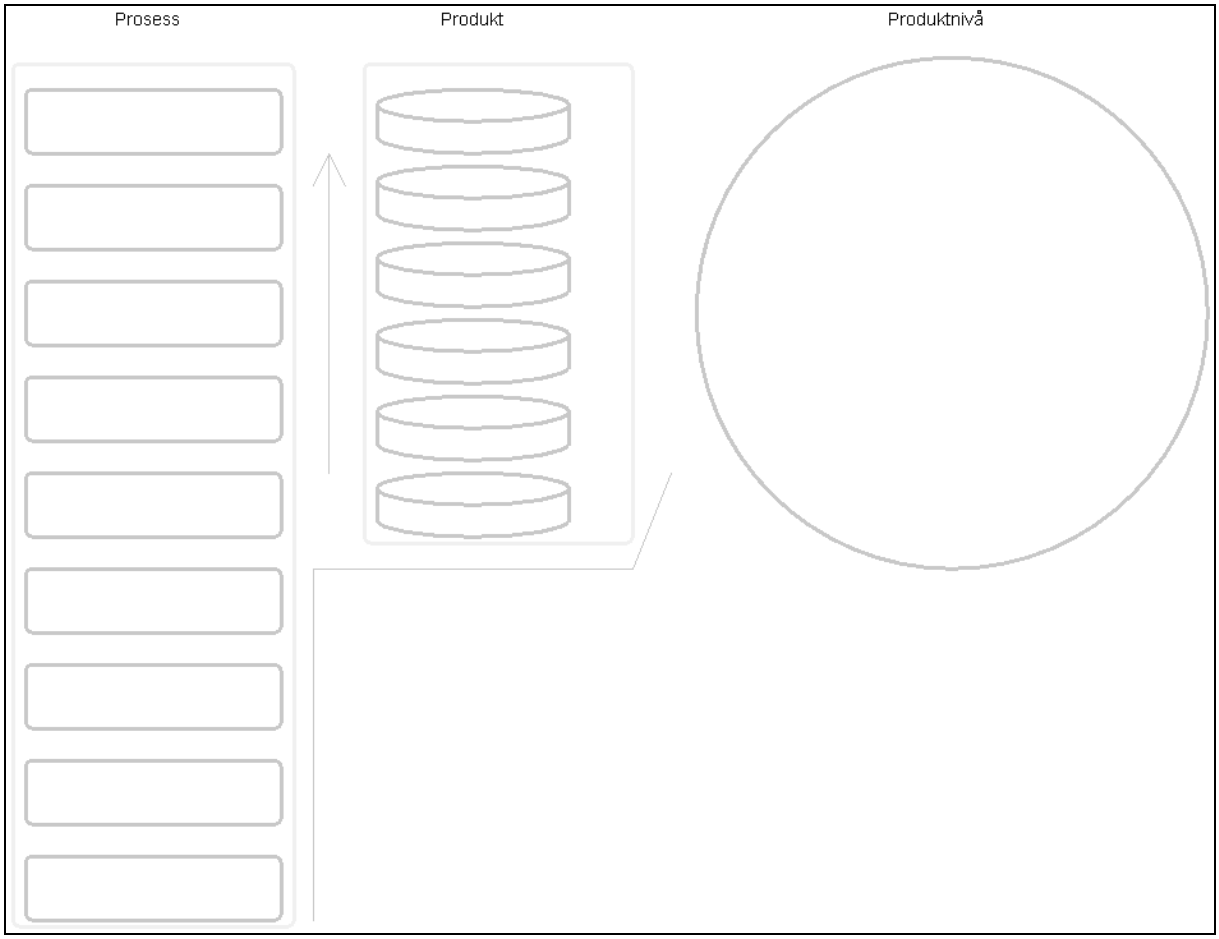
## 4.2 Utvikling

Denne delen skal gi en oversikt over de forskjellige delene prototypen er bygd opp av og hvilke kriterier som er lagt til grunn for valgene.

I utviklingen av prototypen er de fleste komponentstrukturene hentet fra en utviklingspakke i JDK (Java Development Kit) kalt Java2D [Knu99]. Eksempler på komponentstrukturer som er benyttet er RoundRectangle2D, Arc2D, Elllipse2D og Line2D. Noen av designidéene er også basert på generelle konsepter fra datagrafikk [Amm98]. I tillegg er det tatt hensyn til noen informasjons-visualiseringsprinsipper utviklet av Edward R. Tufte [Tuf90, Tuf97, Tuf01]. Disse prinsippene handler om å velge riktig visualiseringsmetode, fremheve viktige data og nedtone mindre viktig grafikk.

### 4.2.1 Brukergrensesnitt

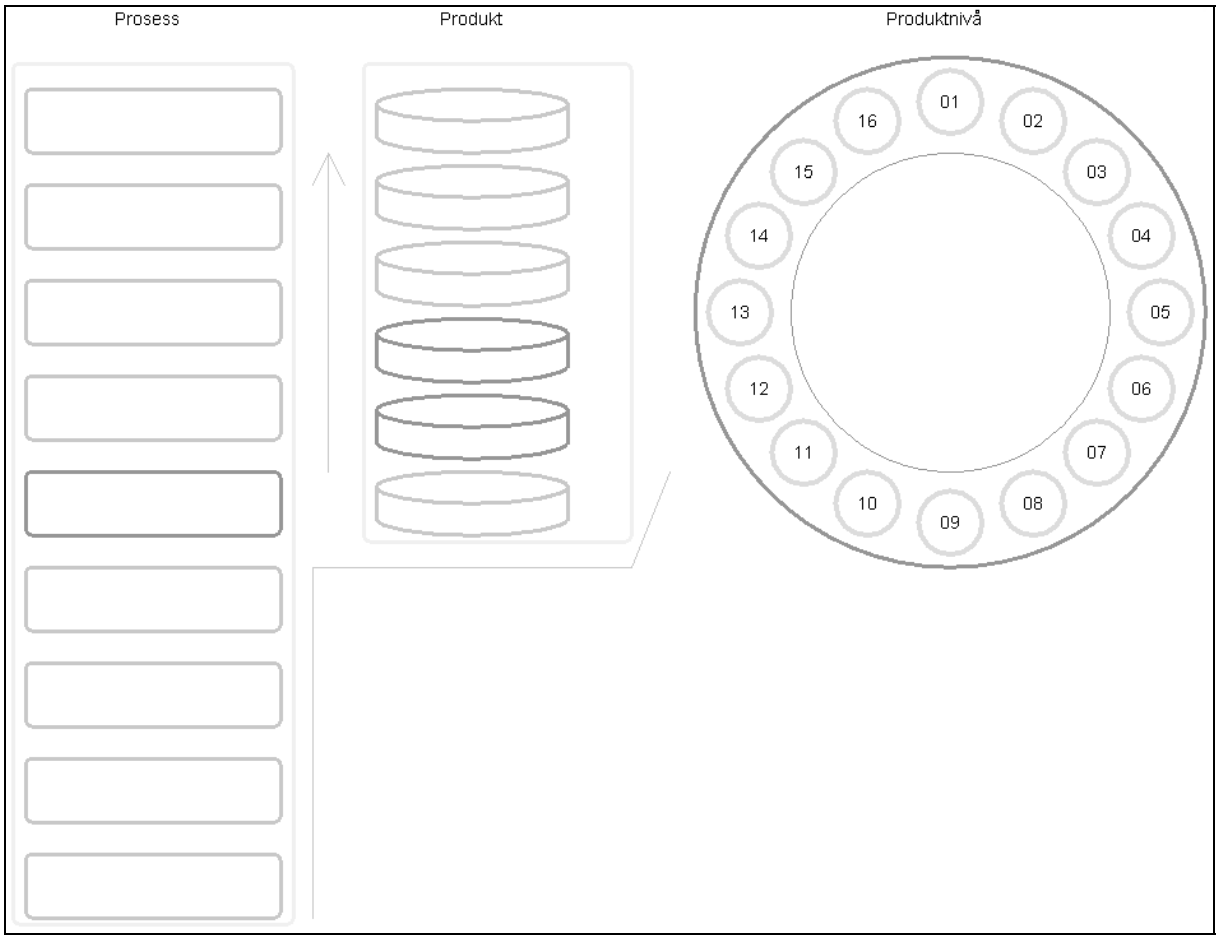
Figur 13 viser prototypens brukergrensesnitt. Den er delt opp i tre separerte bilder; prosess-, produkt- og produktnivåbildet. Prosessbildet er bygd opp med tanke på å gi et enkelt og forståelig bilde av hvordan hovedbestanddelene i prosessen ligger i forhold til hverandre. Produktbildet er bygd opp som en stabel hvor hver enkel del av produktet blir lagt på etter hvert som produktet utvikler seg gjennom produksjonsprosessen. Produktnivået illustrerer nåværende nivå av produktet og den informasjonen som er generert på dette nivået. Det er også mulig å vise ekstra informasjon ved å velge spesifikke produkter internt i nivået (se figur 18). Strukturen for produktnivåbildet er basert på en ringløsningsidé [TM02] for å utnytte den begrensete plassen best mulig. I figur 13 er det også et ledig område nederst til høyre. Dette område blir benyttet til informasjonsvisualisering ved hjelp av parallellkoordinater eller glyphs og blir beskrevet nærmere i kapittel 4.2.3.




**Figur 13 - Prototyp**

### 4.2.2 Koblinger mellom prosess-, produkt- og produktnivåbildet

Figur 14 viser et eksempel på hvordan prosess-, produkt- og produktnivåbildet er koblet sammen. Som illustrert i dette eksempelet, gir et spesifikt valg i prosessbildet to valg i produktbildet, og informasjonen i de to produktnivåene i produktbildet vises i produktnivåbildet. Den innerste sirkelen viser det nederste av de valgte nivåene i produktbildet.



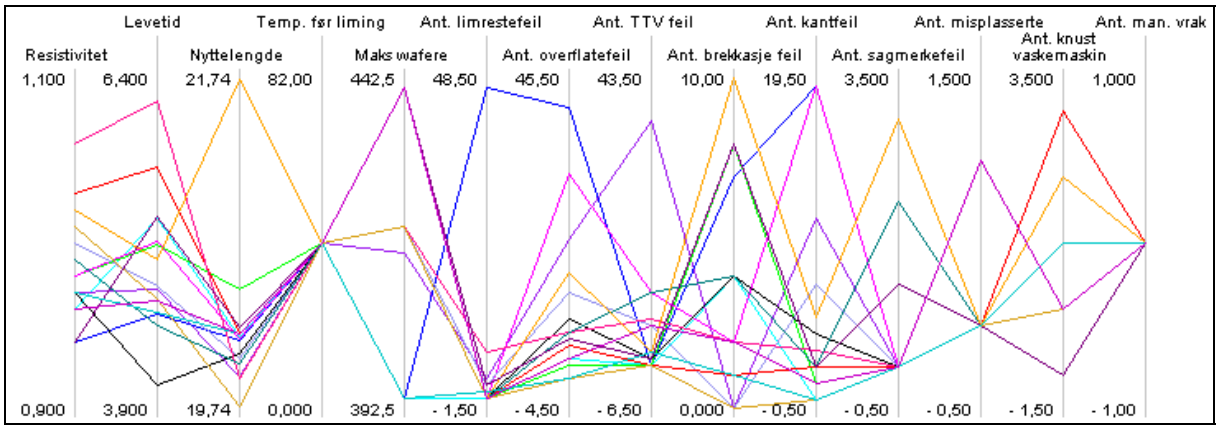

**Figur 14 - Koblinger mellom prosess-, produkt- og produktnivåbildet**

### 4.2.3 Informasjonsvisualisering

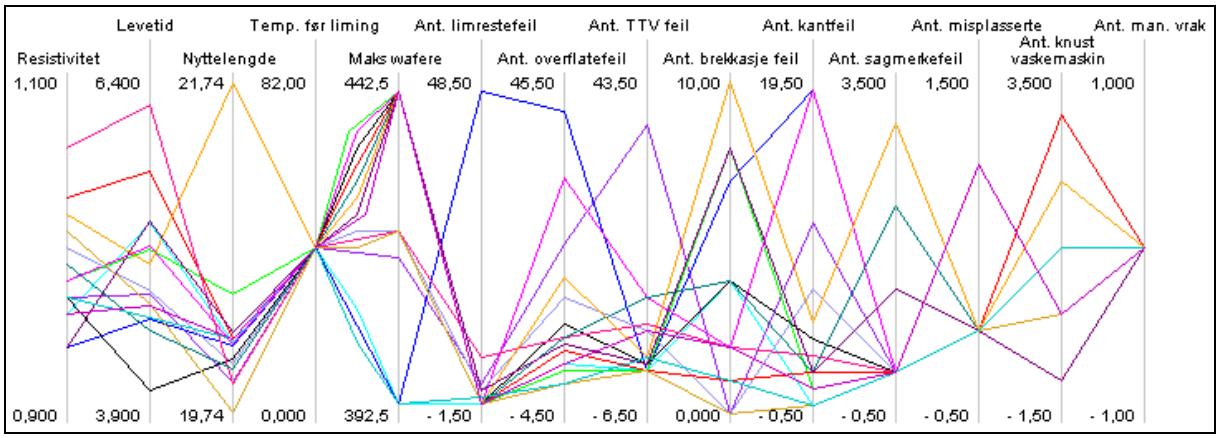
I tillegg til den informasjonen som vises i produktnivåbildet er det også satt av plass til informasjonsvisualisering som skal gjøre det mulig for brukeren av systemet å sammenligne produktene som befinner seg på samme produktnivå. De to konseptene som er implementert er parallellkoordinater [ID90] og glyphs [SF+72]. Disse presentasjonskonseptene ble valgt fordi parallellkoordinater egner seg som analyseverktøy, mens glyphs skal gjøre det mulig for brukerne av systemet og oppdage unormale parameterverdier hurtig i sanntid. De representerer dermed to forskjellige analysegrunnlag i en produksjonsprosess. Dataene som er benyttet her, er hentet fra den produksjonsprosessen som er benyttet som case study i kapittel 5.

**Parallellkoordinater**

Figur 15 viser hvordan hvert produkts parametere illustreres ved hjelp av parallell-koordinater.




**Figur 15 - Parallellkoordinater**

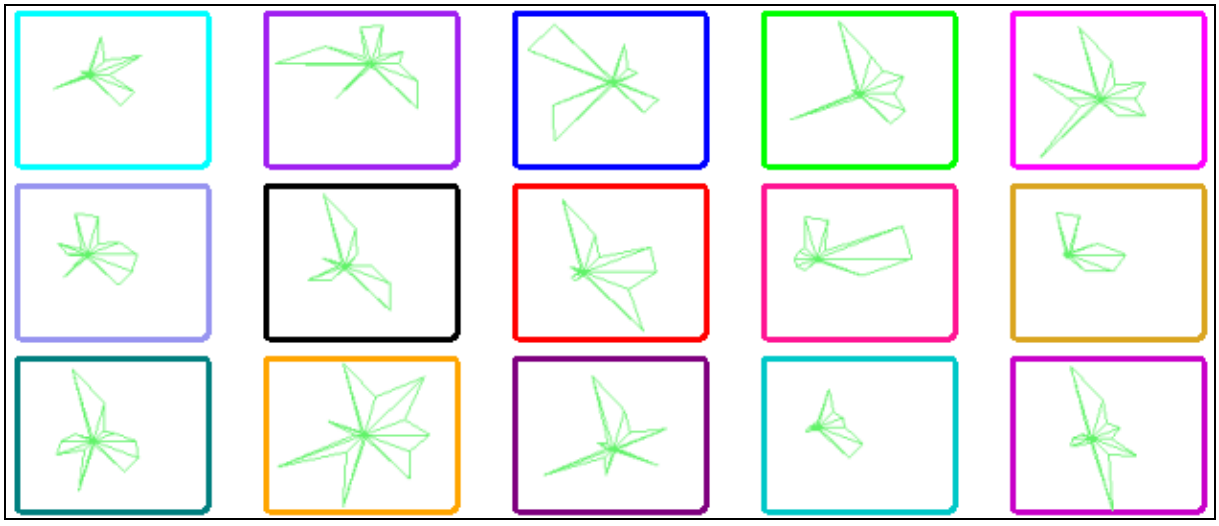



**Figur 16 - Forbedret design av parallellkoordinater**

Problemet med parallellkoordinater er at hvis produkter har like parameterverdier etter hverandre (aktiv parameter og naboparametere) vil disse legge seg på hverandre og en vil dermed ikke kunne følge produktets parameterverdier. Figur 16 viser en mulig løsning på dette problemet. Denne løsningen går ut på å åpne det område det gjelder ved å legge inn et punkt ekstra slik at koblingen mellom parameterne blir synlig. Det hadde kanskje vært bedre å bruke fine kurveelementer [GK03], men dette ville ta en del lenger tid å designe og ble derfor nedprioritert. Hensikten var å vise at det kan oppstå problemer når man benytter parallell-koordinater for å presentere store datamengder.

## Glyphs

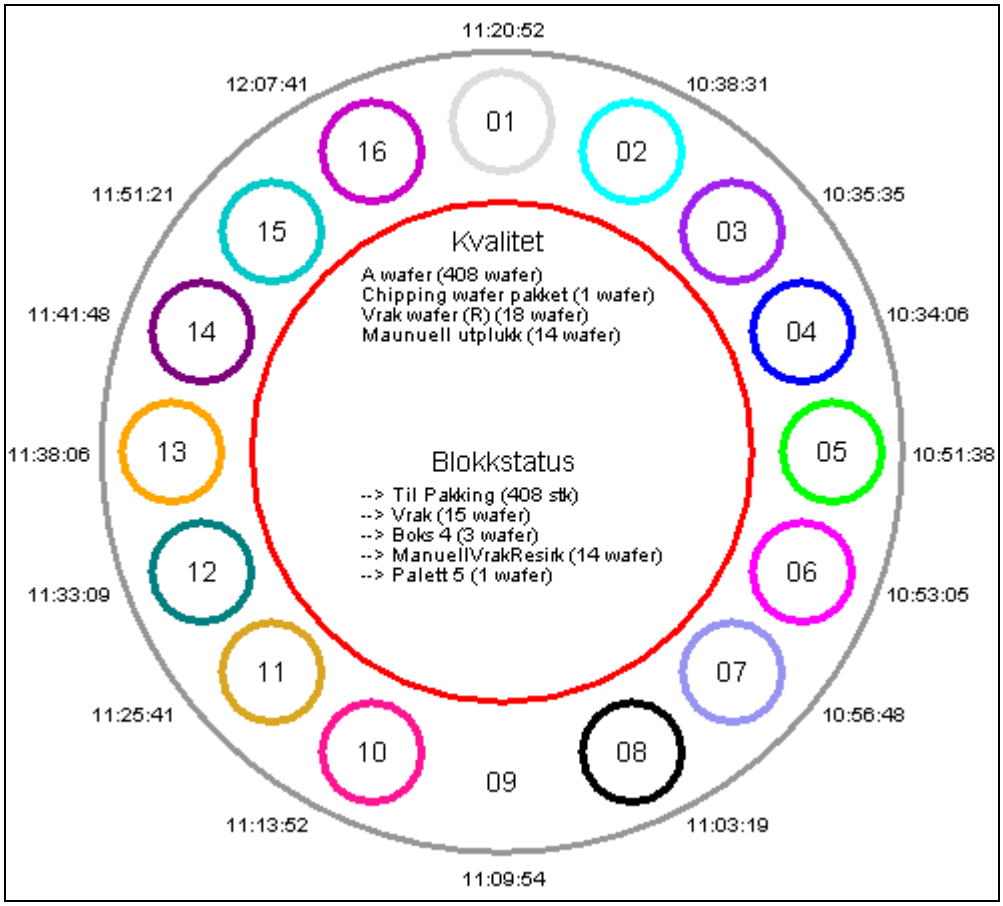
Figur 17 viser en glyphspresentasjon av de samme produktene og hvert enkelt produkts parameterverdier. Det optimale vill ha vært å foreta en bruker-undersøkelse for å teste ut flere glyphstyper, men på grunn av begrenset tid kunne ikke dette gjennomføres.



**Figur 17 - Stjerneglyphs [SF+72]**

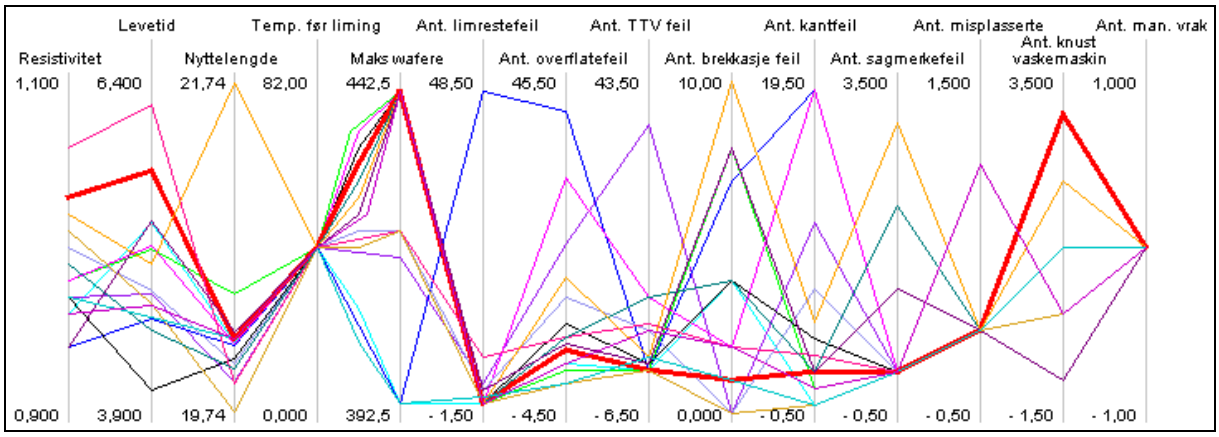
## Valg av produkter på produktnivå

Figur 18 viser hvordan informasjonen til produktet som er valgt vises i sirkelen i midten.




**Figur 18 - Valg av produkt på produktnivå**

I figur 19 blir parallellkoordinatene til produktet som er valgt uthevet.




**Figur 19 - Visualisering av parallellkoordinater ved valgt produkt**

## 4.3 Demonstrasjon og evaluering

Denne delen gir en oversikt over demonstrasjoner og tilbakemeldinger under utviklingen av prototypen.

**Prediktor**

Gruppen hos Prediktor bestod av tre personer. Demonstrasjonen ble utført 02.04.2004.

En tidlig versjon av prototypen ble demonstrert for et lite utvalg av ansatte hos oppdragsgiver. Før demonstrasjonen av prototypen ble et utvalg av visualiserings-teknikker omtalt i kapittel 2, presentert. Dette gav et grunnlag for å vurdere prototypen ut fra forskjellige konsepter. Etter å ha vist prototypens funksjonalitet ble forbedringer diskutert.

I denne versjonen var bare parallellkoordinater implementert. Gruppen mente at dette konseptet kanskje kunne bli litt vanskelig for operatørene i produksjonen hos ScanWafer III å benytte, fordi parallellkoordinater ikke egner seg så godt til analyse av produktenes parameterverdiutvikling i sanntid. Denne presentasjonen kunne fort bli stor, noe som medførte at den lett kunne bli vanskelig å få oversikt over. Det kunne derfor bli vanskelig å håndtere dataene hurtig og effektivt da hensikten med informasjonsvisualisering i sanntid, var å oppdage unormale parameterverdier og ikke nødvendigvis å oppfatte de riktige verdiene på parameterne. Parallell-koordinater var derimot et godt analyseverktøy for å sammenligne produkter på hvert produktnivå. Derfor kunne dette visualiseringskonseptet være et verktøy som prosessingeniørene kunne benytte for etteranalyser av produksjonsprosessen.

Et av konseptene som ble presentert før demonstrasjonen av prototypen var glyphs. En av tilbakemeldingene som kom fra testgruppen var at dette konseptet kanskje var enklere for operatørene hos ScanWafer III å forstå fordi dette gav et godt bilde av unormale parameterverdier. Glyphsutformingen er bygd opp ved at en modell for hvert enkelt produkt ble generert basert på parameterverdiene og ikke illustrert i samme koordinatsystem som var tilfelle for parallellkoordinater.

Etter å ha vurdert tilbakemeldingene fra demonstrasjonen hos oppdragsgiver, ble det implementert mer funksjonalitet i prototypen.

**IFE (Institutt For Energiteknikk)**

Gruppen hos IFE bestod av 2 personer. De har god erfaring med design av brukergrensesnitt, særlig innenfor kjernekraft- og petroleumssektoren, og kan derfor betraktes som eksperter på området. Demonstrasjonen ble utført 23.04.2004.

Som hos Prediktor ble først ulike konsepter for informasjonsvisualisering presentert, for deretter å demonstrere den forbedrede prototypen. At det i denne versjonen var implementert visualisering ved hjelp av glyphs i tillegg til parallell-koordinater, syntes representantene fra IFE var interessante kombinasjoner.

Etter å ha demonstrert prototypen ble forbedringer diskutert. Gruppen syntes prototypen var godt egnet for å visualisere produkt- og produksjonsprosesser, men hadde noen innvendinger mot at farger ble benyttet for tidlig i prosessen. I prototypen som ble presentert fikk produktene i produksjonsprosessen tildelt en farge med en gang de var opprettet. Dette kunne fort forstyrre visualiseringen da hensikten med fargebruken var å relatere hvert enkelt produkt til produktets parametere når informasjonsvisualiseringskonsepter som for eksempel parallellkoordinater ble benyttet. Parametere, som for eksempel produktfeil, ble først registret senere i produksjonsprosessen og det var derfor mer naturlig å tilegne farger på dette tidspunktet fordi informasjonsvisualiseringskonseptene først ble benyttet da. Det at det var benyttet nedtoning av mindre viktige deler av prototypen var bra og i tråd med Edward R. Tuftes prinsipper [Tuf01].

Etter denne gjennomgangen ble prototypen ytterligere justert.

# 5 CASE STUDY

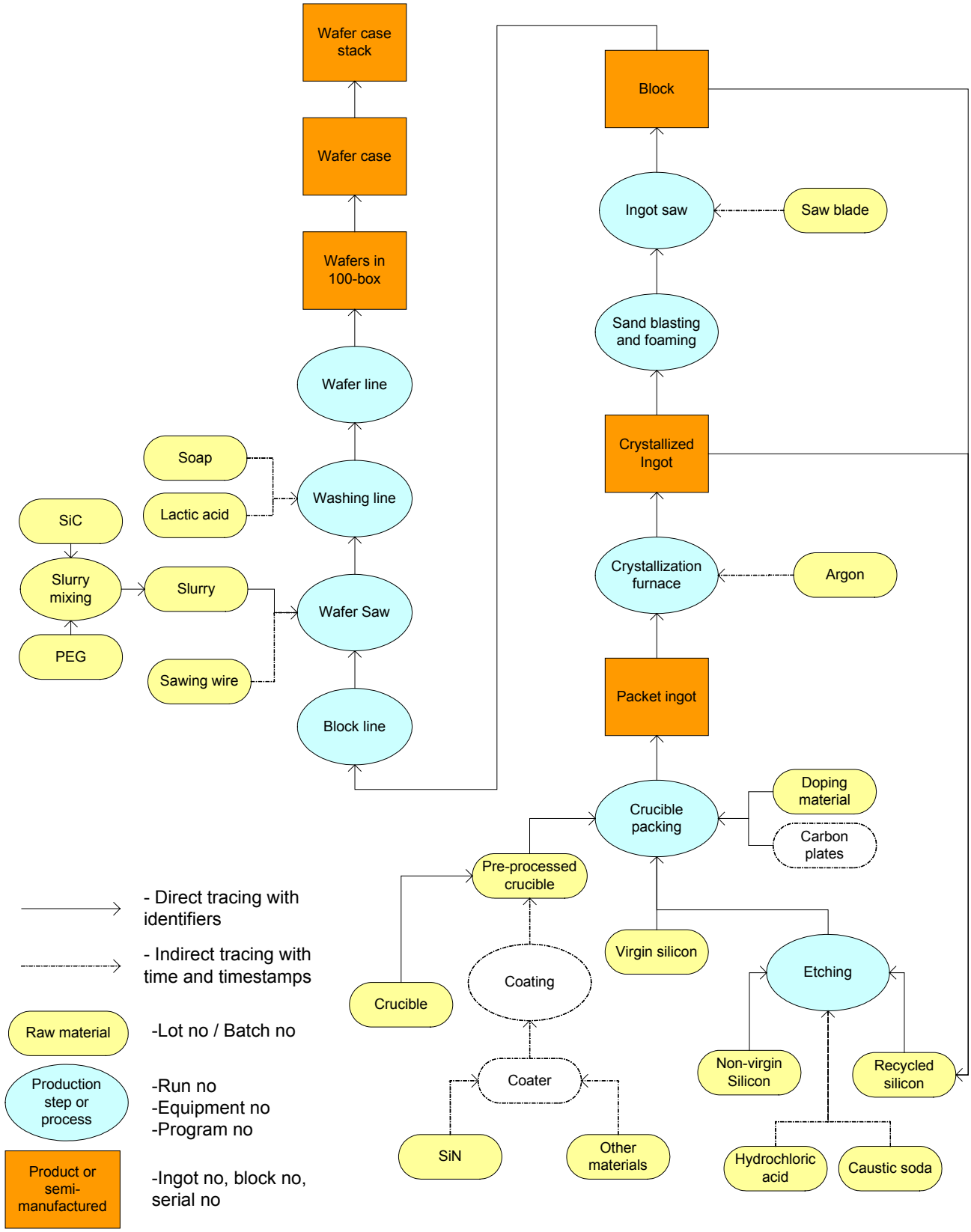
Produksjonsprosessen til ScanWafer III på Herøya i Porsgrunn er valgt som case study. ScanWafer III er en forholdsvis ny bedrift hvor produksjonen startet i 2003. Fabrikken eies av ScanWafer ASA.

## 5.1 Bakgrunn

ScanWafer produserer multikrystallinske silisiumswafere for solcelleindustrien. En wafer er en tynn silisiumsskive med halvlederegenskaper, og er den sentrale delen i en solcelle som omvandler sollys til elektrisitet. Selskapet er en av verdens største produsenter av multikrystalliske wafere. Bedriften har en høyteknologisk metallurgisk virksomhet. Teknologiutvikling er høyt prioritert og skjer i samarbeid med universiteter, utstyrleverandører og kunder. Produksjonen foregår i Glomfjord i Nordland og ved en ny fabrikk (ScanWafer III) på Herøya i Porsgrunn. Verdensmarkedet for solenergi øker med 30-40 prosent per år. ScanWafers hovedmarkeder er Asia (spesielt Japan) og Europa (spesielt Tyskland og Sør-Europa). Bedriften har få men store kunder. Kundeforholdene baseres på langsiktige avtaler. Nesten hele produksjonen eksporteres. ScanWafer er et almennaksjeselskap (ASA) med 277 fast ansatte. Selskapet ble etablert i 1994. Administrasjonssenteret ligger på Høvik utenfor Oslo. [ÅSW03]

Produkt- og prosessflyten ble kartlagt ved først å studere forprosjektrapporten [Fle02] som oppdragsgiver hadde laget for så senere å besøke ScanWafer III for å teste ut de konseptene som skulle visualiseres i en prototyp. Utgangspunktet for dette besøket var å studere produksjonsprosessen og få et bilde av hvordan dataene som ble samlet inn i prosessen ble benyttet. Dette ble gjort ved å observere driften og føre uformelle samtaler med noen av de ansatte om hva de synes om det nåværende systemet og legge frem visse visualiseringsidéer.

Figur 20 beskriver sporingsstrukturen for produksjonen og har blitt brukt som utgangspunkt for å tilpasse prototypen til ScanWafer IIIs produkt- og produksjonsprosess. Produkt- og produksjonsprosessen blir beskrevet i 5.2.




**Figur 20 - Sporingsstrukturen, ScanWafer III [Fle02]**

## 5.2 Produksjonsbeskrivelse

Silikon blir mottatt fra leverandører i forskjellige former og pakninger og blir delt inn i to hoveddeler; ren silikon som kan brukes direkte og uren silikon som renses i en etseprosess før den kan brukes. Deretter blir smeltedigler fylt med silikon og dekket med karbonplater på sidene og i bunnen. Disse blir festet med en metalltråd. De fylte diglene blir deretter plassert i en smelteovn for å smelte og krystallisere. Denne prosessen tar ca. 40 timer.

Ingotblokkene som er silisiumsblokker som dannes etter smelting og krystallisering av silisiumsmetall i ovn, blir deretter satt til kjøling. Etter at ingotblokkene er nedkjølt blir karbonplatene på sidene og i bunnen fjernet og ingotblokken blir sandblåst. Så gjennomføres en manuell kvalitetskontroll. Ved synlige feil på ingot-blokkene blir de kassert og resirkulert.

Ingotblokkene blir deretter festet til en karbonplate for å kunne holde dem under saging. Først blir toppen og de fire sidene kuttet av. Restene fra dette blir resirkulert. Deretter blir ingotblokkene kuttet til mindre blokker og lagt i polyesterbokser for å beskytte dem. En ny manuell inspeksjon blir gjennomført og blokker med synlige sår blir fjernet.

Blokkene blir så plassert på en blokklinje hvor de blir målt, vasket og tørket og nye blokker som ikke tilfredstiller målene blir kassert. Deretter blir hver blokk limt på en glassplate som igjen blir limt på en aluminiumsramme. Denne rammen passer inn i fester på wafersagen hvor blokkene blir saget til wafere. Aluminiumsrammen blir fjernet etter sageprosessen og waferne blir vasket. En ny manuell kontroll blir så gjennomført.

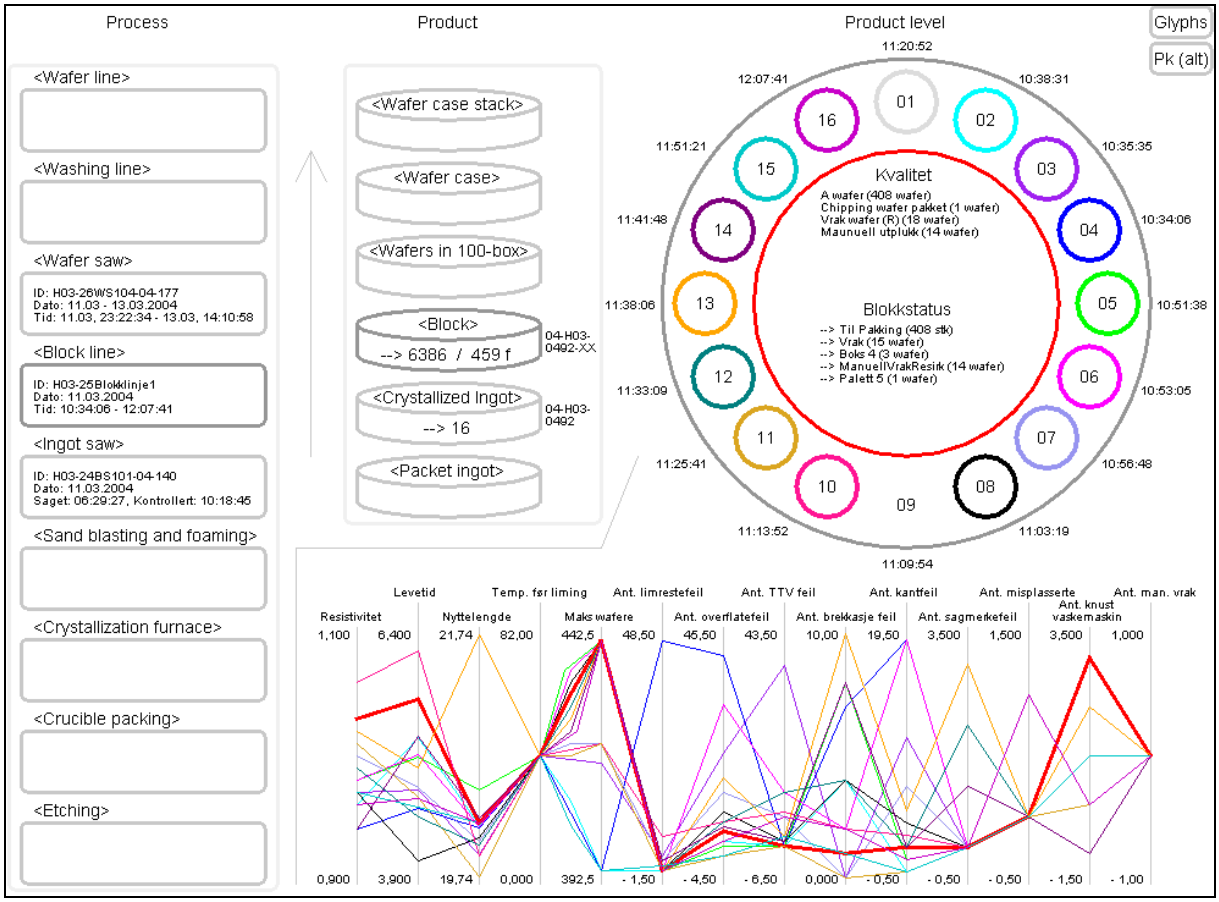
Waferne blir deretter plassert på en waferlinje hvor de blir kontrolleres og pakkes i små bokser. De små boksene blir plassert i større esker som igjen blir plassert på en pall.

Denne produksjonsbeskrivelsen er basert på beskrivelser av produksjonsflyten i forprosjektrapporten [Fle02].

## 5.3 Prototyp

Navn på prosess- og produktssteg er implementert i grensesnittet for prototypen for å tilpasse denne til ScanWafer IIIs produkt- og produksjonsprosess. Produksjonsdata som blir visualisert i prototypen er hentet fra en database som lagrer data direkte fra denne produksjonen.

Figur 21 viser grensesnittet for prototypen med et eksempel, med utfylte verdier fra en rapport, basert på data fra denne produksjonsprosessen. I prototypen er prosess- og produktbilde avskilt i forhold til sporingsstrukturen i figur 20.




**Figur 21 - Prototyp (ScanWafer III)**

## 5.4 Demonstrasjon og evaluering

### 5.4.1 Deltakere

Testpanelet hos ScanWafer III bestod av tre personer (1 prosessingeniør, 1 skiftleder og 1 operatør). Av tidsmessige og praktiske årsaker ble utvalget lite, men var allikevel representativt for brukerne. Demonstrasjonen ble utført 30.04.2004.

### 5.4.2 Gjennomføring av demonstrasjonen

Demonstrasjonen ble utført ved at brukerne først fikk en introduksjon rundt de konseptene som var brukt i prototypen og hvorfor den var bygd opp på den måten den var. Deretter ble prototypen demonstrert med detaljerte forklaringer.

### 5.4.3 Resultater

Denne delen gir en oversikt over de tilbakemeldingene som ble gitt under demonstrasjonen om prototypen og hvilke forbedringspotensialer som ble skissert.

**Dagens situasjon sammenlignet med nye visualiseringsteknikker**

Alle hadde sett at strukturen i dag med store tabeller var vanskelig og lite oversiktelig da lite av informasjonen kom frem på et skjermbilde. En av tilbakemeldingene under introduksjonseksempelet var at det var lett å gå seg bort i rapporttabellene når en skulle finne visse nøkkeltall. Derfor syntes alle at idéen med å visualisere alt på et skjermbilde var god og alle var også med på tankegangen bak konseptene etter å ha blitt forklart hvordan de var bygd opp. Et problem var å se for seg multidimensjonal presentasjon av parameterne da testpanelet som regel var vant til å forholde seg til tidsutviklingen til en parameter, men alle så helt klart en fordel med en slik presentasjon av dataene.

**Informasjonsvisualiseringskonsepter**

Både glyphs og parallellkoordinater ble presentert. Da glyphspresentasjon som regel er enklere å forstå var det et litt uventet resultat at alle forstod sammenhengen mellom parameterne i en parallellkoordinatpresentasjon bedre enn ved glyphs. Det kan ha vært valget av glyphstype, stjerneglyphs [SF+72], som gjorde dette utslaget. Prosessingeniøren hadde lettere for å forstå informasjonsvisualiseringen ved hjelp av glyphs enn skiftlederen og operatøren. Vi diskuterte deretter ulike typer datapresentasjoner basert på glyphs, og tilbakemeldingene fra skiftleder og operatør var at ansiktsglyphs [Che73] kanskje hadde egnet seg bedre.

Siden store deler av operatørarbeidsstokken bestod av personer som ikke hadde gode datakunnskaper, var tilbakemeldingen også at det å innføre for avanserte visualiseringer kunne bli problematisk. Hvis slike visualiseringer skulle innføres, måtte det en grundig treningsperiode til hvor visualiseringen gradvis ble innført.

## Forbedringspotensialer

Følgende områder for forbedring ble berørt:

- Bruk av dynamiske verdier
  - Dynamiske verdier blir benyttet for å utnytte plassen som er satt av for visualisering best mulig, men det kan være forvirrende ved sammenligning av forskjellige produksjoner hvor disse verdiene varierer veldig. En mulig løsning vil kunne være å innføre statiske grenseverdier for akseptable verdier.
- Presentasjon av parameterne for blokkene i to deler i samme bilde
  - Parameterne skal ha både høye og lave verdier (for eksempel antall feil; lave verdier). Dette skaper visuelle forstyrrelser i bildet og kan utbedres ved å dele bildet i to slik at man får ett for høye verdier og ett for lave verdier
- Farger i prototypen
  - Farger kan være vanskelig å oppfatte for eksempel av fargeblinde. Ved å innføre gode filtreringsmuligheter for å fjerne en del unødvendig data kan fargebruken sannsynligvis reduseres.
- Glyphspresentasjon
  - Hvis man ikke har god erfaring med glyphs kan disse være vanskelige å tolke. Ved å innføre glyphs med optimale verdier som ligger bakenfor resultatglyphs kan avvik fra de optimale verdiene lettere oppdages.
- Navigering i prosessen
  - For hele tiden å vite hvilken enhet i prosessen som er aktiv, ville det vært fordelaktig å innføre et tekstfelt øverst i prototypen som angår dette.

## Oppsummering

Demonstrasjonen viste at denne måten å presentere data på hadde merkbare fordeler sammenlignet med eksisterende metoder fordi man fikk bedre oversikt ved at dataene ble presentert i ett skjermbilde. Dette gav større muligheter til å oppdage feil og treffe tiltak ved eventuelle avvik fra kvalitetstandarder og derigjennom et godt grunnlag for å øke kvaliteten på produserte enheter.

# 6 KONKLUSJON OG FREMTIDIGE UTFORDRINGER

## 6.1 Konklusjon

I litteraturstudiet ble det forklart at visualisering av data har en lang historie og at det eksisterer mange konsepter for dette. Problemet med en del av disse konseptene er at de er utviklet for å løse ett spesifikt problem og at de derfor ikke er direkte anvendbare i andre sammenhenger. Basert på helheten i litteraturstudiet gav allikevel konseptene et godt grunnlag for å designe en prototyp med den hensikt å vise metoder for effektiv visualisering av data i en produksjonsprosess.

Utfordringene i designet av prototypen var å tilpasse noen av disse konseptene til produksjonsdatavisualisering på en slik måte at det gav klare visuelle forbedringer i forhold til den løsningen som ble benyttet i dag.

I samtaler med oppdragsgiver og IFE ble forskjellige konsepter diskutert for å komme frem til visualiseringsteknikker som kunne illustrere produksjonsdata fra en reell produksjonsprosess og som kunne implementeres i prototypen. Denne ble så demonstrert for et utvalg brukere av denne produksjonsprosessen.

Som nevnt innledningsvis har denne oppgaven ikke hatt som formål å finne den optimale presentasjonsformen, men å vise at alternative konsepter kan gi mer hensiktsmessige presentasjoner. Tilbakemeldinger underveis i utviklingen av prototypen og demonstrasjonen for en testgruppe hos ScanWafer III gav som resultat, indikasjoner på at visualiseringskonseptene som for eksempel parallell-koordinater eller glyphs egnet seg bra for visualisering av produksjonsdata.

Fordeler ved disse visualiseringskonseptene er at de gir muligheter for presentasjon av store og komplekse datamengder på liten skjermflate. De gir også oversiktlige grafer/figurer som gir bedre grunnlag for å treffe riktige og hurtige tiltak ved avvik fra kvalitetsstandarder. Til slutt gir visualisering av produkt- og produksjons-prosesstrukturen muligheter for rask traversering. Disse fordelene gjør det enklere å spore hvor feil og kvalitetsforringelser oppstår.

Prototypen har klare forbedringspotensialer som det er relativt enkelt å implementere.

Forutsetningen for bruk er at brukerne gis god opplæring i systematikk og virkemåte for parallellkoordinater og glyphs.

## 6.2 Videre arbeid og fremtidige utfordringer

Videre arbeid og utfordringer vil være å teste ut de eksisterende konseptene for trestrukturer, prosesstopologier og informasjonsvisualisering, som allerede ligger i prototypen, på andre produkt- og produksjonsprosesser for å undersøke om disse konseptene kan benyttes der eller om andre konsepter fungerer bedre for disse.

En videreutvikling av den eksisterende prototypen bør også gjøres med fokus på bedre bruk av fargekoder og nye visualiseringskonsepter. Dette vil gjøre prototypen mer fleksibel for testing og evaluering under nye demonstrasjoner. Det kan også være interessant å se på muligheten for å inkludere flere forskjellige designidéer innenfor samme konsept. Dette gjelder spesielt for glyphs hvor det finnes mange alternativer [War02].

Filtreringsmekanismer kan også være et interessant emne å utforske for å bedre visualiseringen av de store og komplekse datamengdene. Dette vil gjøre det mulig å visualisere enda større datasett ved å fjerne data som ikke er beslutningsgivende.

# 7 REFERANSER


[AH98] Keith Andrews, Helmut Heidegger
«Information Slices: Visualising and Exploring Large Hierarchies using Cascading, Semi-Circular Discs» i
*InfoVis'98: Proceedings of the 1998 IEEE Symposium on Information Visualization, Late Breaking Hot Topic Paper*
Oktober 1998 (Elektronisk, ftp: IICM, Østerrike)
ftp://ftp.iicm.edu/pub/papers/ivis98.pdf (April 2004)

[Amm98] Leen Ammeraal
*Computer Graphics for Java Programmers*
John Wiley & Sons, 1998

[Asi85] Daniel Asimov
«The Grand Tour: A Tool for Viewing Multidimensional Data» i
*SIAM Journal on Scientific and Statistical Computing*
Vol. 6, Nr. 1, Januar 1985, ss. 128–143

[BC87] Richard A. Becker and William S. Cleveland
«Brushing Scatterplots» i
*Technometrics*
Vol. 29, Nr. 2, Mai 1987, ss. 127–142

[Bed90] Jeff Beddow
«Shape Coding of Multidimensional Data on a Microcomputer Display» i
*Visualization '90: Proceedings of the 1st IEEE Conference on Visualization*
1990, ss. 238-246

[BWV03] Alf Ove Braseth, Robin Welch, Øystein Veland
*A Building Block for Information Rich Displays*
Tilgjengelig fra IFE, Halden
Presentert på IFEA-konferansen for «alarmhåntering», Gardermoen, 25-26 Nov. 2003

[Che73] Herman Chernoff
«The Use of Faces to Represent Points in k-Dimensional Space Graphically» i
*Journal of American Statistical Association*
Vol. 68, Nr. 342, Juni 1973, ss. 361-368

[Cle93] William S. Cleveland
*Visualizing Data*
Hobart Press, 1993

[CR75] Herman Chernoff, M. Haseeb Rizvi
«Effect on Classification Error or Random Permutations of Features in Representing Multi-variate Data by Faces » i
*Journal of American Statistical Association*
Vol. 70, Nr. 351, September 1975, ss. 548-554

[CR96] Mei C. Chuah, Steven F. Roth
«On the Semantics of Interactive Visualizations» i
*InfoVis'96: Proceedings of the 1996 IEEE Symposium on Information Visualization*
Oktober 1996, ss. 29-36 (elektronisk database: CiteSeer)
http://citeseer.ist.psu.edu/256297.html (April 2004)

[DC01] Russ M. Dabbas, Hung-Nan Chen
«Mining Semiconductor Manufacturing Data for Productivity Improvement – An Integrated Relational Database Approach» i
*Computers in Industry*
Vol. 45, Utgave 1, Mai 2001 (Elektronisk, Database: ScienceDirect)

[Fek03] Jean-Daniel Fekete
«The InfoVis Toolkit» i
Rapport de Recherche de l'INRIA
Nr. 4818, Mai 2003 (Elektronisk, L'INRIA, Frankrike)
http://www.inria.fr/rrrt/rr-4818.html (Februar 2004)
Verktøyets hjemmeside:
http://www.lri.fr/~fekete/InfovisToolkit (April 2004)

[Fle02] Erling Fledsberg
*Pre-study Report: Process Data System for ScanWafer III*
Tilgjengelig fra Prediktor AS, Fredrikstad

[Fur99] George W. Furnas
«The Fisheye View: A New Look at Structured Files» i
*Readings in Information Visualization - Using Vision to Think*
1999, ss. 312-330 (Publisert: Morgan Kaufmann Publishers)
Tidligere Publisert i
*Bell Laboratories Technical Memorandum*
#81-11221-9, 12 Oktober 1981

[FWR99] Ying-Huey Fua, Matthew O. Ward, Elke A. Rundensteiner
«Hierarchical Parallel Coordinates for Exploration of Large Datasets» i
*Visualization '99: Proceedings of the IEEE Conference on Visualization*
Oktober 1999, ss. 43-50 og 508

[GK03] Martin Graham, Jessie Kennedy
«Using Curves to Enhance Parallel Coordinate Visualizations» i
*IV '03: Proceedings of 7th IEEE Conference on Information Visualization*
Juli 2003, ss 10-16

[Goo03] John Goodall
"Software" i
*Information Visualization*
November 2003 (Elektronisk dokument)
http://iv.homeunix.org/software.php (Februar 2004)

[GT87] Jonathan L. Gross, Thomas W. Tucker
*Topological Graph Theory*
Wiley Interscience Series in Discrete Mathematics and Optimalization, Wiley, 1987

[HK97] Marti A. Hearst, Chandu Karadi
«Cat-a-Cone: An Interactive Interface for Specifying Searches and Viewing Retrieval Results using a Large Category Hierarchy» i
*Proceedings of SIGIR-97: 20th ACM Conference on Research and Development in Information Retrieval*
1997, ss. 246-255 (Elektronisk, Database: CiteSeer)
http://citeseer.ist.psu.edu/hearst97catcone.html (Mars 2004)

[ID90] Alfred Inselberg, Bernard Dimsdale
«Parallel Coordinates: A Tool for Visualizing Multi-Dimensional Geometry» i
*Visualization '90: Proceedings of the 1st IEEE Conference on Visualization*
1990, ss. 361-378

[Ins99] Alfred Inselberg
«Multidimensional Detective» i
*Readings in Information Visualization - Using Vision to Think*
1999, ss. 107-114 (Publisert: Morgan Kaufmann Publishers)
Tidligere Publisert i
*InfoVis'97: Proceedings of the 1997 IEEE Symposium on Information Visualization*
Oktober 1997, ss. 100-107

[Jon98] Christopher V. Jones
"Visualization and Optimization" i
*Interactive Transactions of OR/MS*
September 1998 (Elektronisk dokument)
http://catt.bus.okstate.edu/jones98/index.html (Mars 2004)

[KE01] Martin Kraus, Thomas Ertl
«Interactive Data Exploration with Customized Glyphs» i
WSCG 2001 Conference Proceedings
2001 (Elektronisk, Database: CiteSeer)
http://citeseer.ist.psu.edu/kraus01interactive.html (April 2004)

[KH+02] Daniel A Keim, Ming C Hao, Umesh Dayal, Meichun Hsu
«Pixel Bar Charts: a Visualization Technique for Very Large Multi-Attribute Data Sets» i
*Information Visualization*
Vol. 1, Nr. 1, Mars 2002, ss. 20-34 (Elektronisk, Palgrave Macmillan)
http://www.palgrave-journals.com/ivs/journal/v1/n1/index.html (Januar 2004)

[Knu97] Donald E. Knuth
*The Art of Computer Programming, Volume 1 – Fundamental Algorithms*
Addison-Wesley, 3. utg., 1997

[Knu99] Jonathan Knudsen
*Java 2D Graphics*
O'Reilly, Mai 1999

[KP+03] Daniel A. Keim, Christian Panse, Jörn Schneidewind, Mike Sips, M. C. Hao, U. Dayal
«Pushing the Limit in Visual Data Exploration: Techniques and Applications» i
*Lecture Notes in Computer Science*
Vol. 2821, September 2003, ss. 37-51 (Elektronisk, Database: SpringerLink)

[LWW90] Jeffrey LeBlanc, Matthew O. Ward, Norman Wittels
«Exploring N-Dimensional Databases» i
*Visualization '90: Proceedings of the 1st IEEE Conference on Visualization*
1990, ss. 230-237

[MH98] Guy Melançon, Ivan Herman
«Circular Drawings of Rooted Trees» i
*Reports of the National Research Institute for Mathematics and Computer Science, Amsterdam (NL), Information Systems Report*
INS-R9817, Desember 1998 (Elektronisk, CWI Library Amsterdam)
http://www.cwi.nl/ftp/CWIreports/INS/INS-R9817.pdf (Februar 2004)

[MHM00a] M. Scott Marshall, Ivan Herman, Guy Melançon
«An Object-Oriented Design for Graph Visualization» i
*Reports of the National Research Institute for Mathematics and Computer Science, Amsterdam (NL), Information Systems Report*
INS-R0001, Januar 2000 (Elektronisk, CWI Library Amsterdam)
http://www.cwi.nl/ftp/CWIreports/INS/INS-R0001.pdf (Februar 2004)

[MHM00b] M. Scott Marshall, Ivan Herman, Guy Melançon
«Automatic Generation of Interactive Overview Diagrams for the Navigation of Large Graphs» i
*Reports of the National Research Institute for Mathematics and Computer Science, Amsterdam (NL), Information Systems Report*
INS-R0014, 2000 (Elektronisk, CWI Library Amsterdam)
http://www.cwi.nl/ftp/CWIreports/INS/INS-R0014.pdf (Februar 2004)

[NH03] Quang Vinh Nguyen, Mao Lin Huang
«A Fast Focus + Context Viewing Techniques for the Navigation of Classical Hierarchical Layout» i
*IV '03: Proceedings of 7th IEEE Conference on Information Visualization*
Juli 2003, ss. 42-46

[RC94] Ramana Rao, Stuart K. Card
«The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus+Context Visualization for Tabular Information» i
*Proceedings of the SIGCHI conference on Human factors in computing systems*
1994, ss. 318-322 (Elektronisk, Database: CiteSeer)
http://citeseer.ist.psu.edu/rao94table.html (April 2004)

[RL+96] Steven F. Roth, Peter Lucas, Jeffrey A. Senn, Cristina C. Gomberg, Michael B. Burks, Philip J. Stroffolino, John A. Kolojejchick, Carolyn Dunmire
«Visage: A User Interface Environment for Exploring Information» i
*InfoVis'96: Proceedings of the 1996 IEEE Symposium on Information Visualization*
1996, ss. 3-12 (Elektronisk, Database: CiteSeer)
http://citeseer.ist.psu.edu/roth96visage.html (April 2004)

[RMC91] G. Robertson, J. Mackinlay, and S. Card.
«Cone trees: Animated 3d visualization of hierarchical information» i
*Proc. Of Computer-Human Interaction '91*
1991, ss. 189-194

[RT81] E. M. Reingold, J. S. Tilford
«Tidier Drawing of Trees» i
*IEEE Transactions on Software Engineering*
Vol. 7, Nr. 2, Mars 1981, ss. 223-228

[SBB96] Michael Spenke, Christian Beilken, Thomas Berlage
«FOCUS: The Interactive Table for Product Comparison and Selection» i
*ACM Symposium on User Interface Software and Technology*
1996, ss. 41-50 (Elektronisk, Database: CiteSeer)
http://citeseer.ist.psu.edu/spenke96focus.html (April 2004)

[SF+72] John H. Siegel, Edward J. Farrell, Roger M. Goldwyn, Herman P. Friedman
«The Surgical Implication of Physiologic Patterns in Myocardial Infarction Shock» i
*Surgery*
Vol. 72, Nr. 1, Juli 1972, ss. 126-141

[Shn92] Ben Shneiderman
«Tree visualization with Tree-maps: A 2-d space-filling approach» i
*ACM Transactions on Graphics*
Vol. 11, Nr. 1 , Januar 1992, ss 92-99 (Elektronisk, Database: CiteSeer)
http://citeseer.ist.psu.edu/shneiderman91tree.html (Mars 2004, utkast)

[TJ92] David Turo, Brian Johnson
«Improving the Visualization of Hierarchies with Treemaps: Design Issues and Experimentation» i
*Visualization '92: Proceedings of the IEEE Conference on Visualization*
Oktober 1992, ss. 124-131 (Elektronisk, Database: CiteSeer)
http://citeseer.ist.psu.edu/turo92improving.html (Mars 2004)

[TM02] Soon Tee Teoh, Kwan-Liu Ma
«RINGS: A Technique for Visualizing Large Hierarchies» i
*Lecture Notes in Computer Science*
Vol. 2528, Januar 2002, ss. 268-275 (Elektronisk, Database: SpringerLink)

[Tuf90] Edward R. Tufte
*Envisioning Information*
Graphics Press, 1990

[Tuf97] Edward R. Tufte
*Visual Explanations – Images and Quantities, Evidence and Narrative*
Graphics Press, 1997

[Tuf01] Edward R. Tufte
*The Visual Display of Quantitative Information*
Graphics Press, 2. Utg., 2001

[Tuk77] John W. Tukey
*Exploratory Data Analysis*
Addison-Wesley, 1977

[vLa01] D. L. Van Laar
«Psychological and Cartographic Principles for the Production of Visual Layering Effects in Computer Displays» i
*Displays*
Vol. 22, Utg. 4, September 2001, ss. 125-135 (Elektronisk, Database: ScienceDirect)

[vLD02] Darren Van Laar, Ofer Deshe
«Evaluation of a Visual Layering Methodology for Colour Coding Control Room Displays» i
*Applied Ergonomics*
Vol. 33, Utg. 4, November 2002, ss. 371-377 (Elektronisk, Database: ScienceDirect)

[War94] Matthew O. Ward
«XmdvTool: Integrating Multiple Methods for Visualizing Multivariate Data» i
*Visualization '94: Proceedings of the IEEE Conference on Visualization*
Oktober 1994, ss. 326-333 (Elektronisk, XmdvTools hjemmeside)
http://davis.wpi.edu/~xmdv/doc_vis94.html (Februar 2004)
Verktøyets hjemmeside:
http://davis.wpi.edu/~xmdv/ (April 2004)

[War02] Matthew O. Ward
«A Taxonomy of Glyph Placement Strategies for Multidimensional Data Visualization» i
*Information Visualization*
Vol. 1, Nr. 3/4, Desember 2002, ss 194-210 (Elektronisk, Palgrave Macmillan)
http://www.palgrave-journals.com/ivs/journal/v1/n3/index.html (Januar 2004)
(Elektronisk fritt tilgjengelig via XmdvTools hjemmeside)
http://davis.wpi.edu/~xmdv/docs/jinfovis02_glyphpos.pdf (April 2004, utkast)

[WB97] Pak Chung Wong, R. Daniel Bergeron
«30 Years of Multidimensional Multivariate Visualization» i
*Scientific Visualization - Overviews, Methodologies and Techniques*
1997, ss. 3-33 (Elektronisk, Database: CiteSeer)
http://citeseer.ist.psu.edu/wong97years.html (April 2004)

[WCJ98] Ulrika Wiss, David Carr, Håkan Jonsson
«Evaluating Three-Dimensional Information Visualization Designs: a Case Study of Three Designs» i
*IV '98: Proceedings of 1998 IEEE Conference on Information Visualization*
Juli 1998, ss. 137-144 (Elektronisk, Database: CiteSeer)
http://citeseer.ist.psu.edu/wiss98evaluating.html (Mars 2004)

[WP+02] Colin Ware, Helen Purchase, Linda Colpoys, Matthew McGill
«Cognitive Measurements of Graph Aesthetics» i
*Information Visualization*
Vol. 1, Nr. 2, Juni 2002, ss 103-110 (Elektronisk, Palgrave Macmillan)
http://www.palgrave-journals.com/ivs/journal/v1/n2/index.html (Januar 2004)

[ÅSW03] Scanwafer
*Årsrapport 2003*
Tilgjengelig fra ScanWafer ASA, Høvik

# VEDLEGG A: RÅDATA OG UTREGNINGER

Her følger en oversikt over verdier og utregninger som er brukt i prototypen.

Dataene i tabell 1 til 4 beskriver verdiene for parallellkoordinatene og glyphs. Y-verdiene er en omregning av verdiene for å plassere koordinatene riktig i prototypen.

**Forklaringer til tabell 1 til 4:**
*størst:* høyeste verdi (brukes som verdier for intervallet)
*minst:* minste verdi (brukes som verdier for intervallet)
*total:* sum av verdiene
*gjennomsnitt:* gjennomsnittsverdien
*intervall:* intervallet mellom høyeste og laveste verdi
*pixl./int. 1 (pikselverdi/intervall):* fordelingen av intervallet innenfor det pikselområdet som er tilgjengeling
*endret:* endring av intervall for å gjøre det lettere å plassere disse i prototypen
*fra og til:* nye verdier for størst (fra) og minst (til)
*pixl./int. 2:* nye verdier for fordelingen av intervallet innenfor pikselområdet som er tilgjengelig

**Tabell 1 - Verdier og omregning for resistivitet, levetid, nyttelengde og temperatur før liming**

| Resistivitet | y-verdi | Levetid | y-verdi | Nyttelengde | y-verdi | Temperatur før liming | y-verdi | Produkt nr. |
|---|---|---|---|---|---|---|---|---|
| 0,96 | 685,00 | 5,34 | 629,80 | 20,17 | 702,00 | 41,00 | 645,00 | 2 |
| 0,97 | 675,00 | 4,80 | 673,00 | 19,93 | 726,00 | 41,00 | 645,00 | 3 |
| 0,94 | 705,00 | 4,61 | 688,20 | 20,15 | 704,00 | 41,00 | 645,00 | 4 |
| 0,98 | 665,00 | 5,13 | 646,60 | 20,46 | 673,00 | 41,00 | 645,00 | 5 |
| 0,98 | 665,00 | 5,16 | 644,20 | 20,16 | 703,00 | 41,00 | 645,00 | 6 |
| 1,00 | 645,00 | 4,83 | 670,60 | 20,04 | 715,00 | 41,00 | 645,00 | 7 |
| 0,97 | 675,00 | 4,08 | 730,60 | 20,07 | 712,00 | 41,00 | 645,00 | 8 |
| 1,03 | 615,00 | 5,72 | 599,40 | 20,19 | 700,00 | 41,00 | 645,00 | 9 |
| 1,06 | 585,00 | 6,22 | 559,40 | 19,92 | 727,00 | 41,00 | 645,00 | 10 |
| 1,01 | 635,00 | 4,73 | 678,60 | 19,75 | 744,00 | 41,00 | 645,00 | 11 |
| 0,99 | 655,00 | 4,52 | 695,40 | 20,01 | 718,00 | 41,00 | 645,00 | 12 |
| 1,02 | 625,00 | 5,02 | 655,40 | 21,73 | 546,00 | 41,00 | 645,00 | 13 |
| 0,94 | 705,00 | 5,35 | 629,00 | 20,23 | 696,00 | 41,00 | 645,00 | 14 |
| 0,97 | 675,00 | 4,62 | 687,40 | 20,19 | 700,00 | 41,00 | 645,00 | 15 |
| 0,96 | 685,00 | 4,71 | 680,20 | 20,19 | 700,00 | 41,00 | 645,00 | 16 |
| | | | | | | | | |
| 1,06 | | 6,22 | | 21,73 | | 41,00 | | størst |
| 0,94 | | 4,08 | | 19,75 | | 41,00 | | minst |
| | | | | | | | | totalt |
| | | | | | | | | |
| 0,99 | | 4,99 | | 20,21 | | 41,00 | | gjennomsnitt |
| | | | | | | | | |
| 0,12 | | 2,14 | | 1,98 | | 0,00 | | intervall |
| 1666,67 | | 93,46 | | 5,05 | | | | pixl./int. (1) |
| 0,20 | | 2,50 | | 2,00 | | | | endret |
| | | | | | | | | |
| 1,10 | | 6,40 | | 21,74 | | 82,00 | | fra |
| 0,90 | | 3,90 | | 19,74 | | 0,00 | | til |
| | | | | | | | | |
| 1000 | | 80 | | 100 | | | | pixl./int. (2) |

**Tabell 2 - Verdier og omregning for maks wafere, ant. limrestefeil, ant. overflatefeil og ant. TTV feil**

| Maks wafere | y-verdi | Antall limrestefeil | y-verdi | Antall overflatefeil | y-verdi | Antall TTV feil | y-verdi | Produkt nr. |
|---|---|---|---|---|---|---|---|---|
| 394,00 | 739,00 | 0,00 | 739,00 | 3,00 | 715,00 | 0,00 | 719,00 | 2 |
| 416,00 | 651,00 | 3,00 | 727,00 | 21,00 | 643,00 | 37,00 | 571,00 | 3 |
| 394,00 | 739,00 | 47,00 | 551,00 | 41,00 | 563,00 | 0,00 | 719,00 | 4 |
| 441,00 | 551,00 | 0,00 | 739,00 | 2,00 | 719,00 | 0,00 | 719,00 | 5 |
| 441,00 | 551,00 | 0,00 | 739,00 | 31,00 | 603,00 | 11,00 | 675,00 | 6 |
| 420,00 | 635,00 | 2,00 | 731,00 | 13,00 | 675,00 | 6,00 | 695,00 | 7 |
| 441,00 | 551,00 | 0,00 | 739,00 | 9,00 | 691,00 | 1,00 | 715,00 | 8 |
| 441,00 | 551,00 | 0,00 | 739,00 | 5,00 | 707,00 | 0,00 | 719,00 | 9 |
| 420,00 | 635,00 | 7,00 | 711,00 | 7,00 | 699,00 | 7,00 | 691,00 | 10 |
| 420,00 | 635,00 | 0,00 | 739,00 | 0,00 | 727,00 | 0,00 | 719,00 | 11 |
| 441,00 | 551,00 | 0,00 | 739,00 | 7,00 | 699,00 | 11,00 | 675,00 | 12 |
| 441,00 | 551,00 | 0,00 | 739,00 | 16,00 | 663,00 | 2,00 | 711,00 | 13 |
| 441,00 | 551,00 | 2,00 | 731,00 | 6,00 | 703,00 | 1,00 | 715,00 | 14 |
| 394,00 | 739,00 | 1,00 | 735,00 | 0,00 | 727,00 | 2,00 | 711,00 | 15 |
| 441,00 | 551,00 | 0,00 | 739,00 | 3,00 | 715,00 | 6,00 | 695,00 | 16 |
| | | | | | | | | |
| 441,00 | | 47,00 | | 41,00 | | 37,00 | | størst |
| 394,00 | | 0,00 | | 0,00 | | 0,00 | | minst |
| 6386,00 | | 62,00 | | 164,00 | | 84,00 | | totalt |
| | | | | | | | | |
| 425,73 | | 4,13 | | 10,93 | | 5,60 | | gjennomsnitt |
| | | | | | | | | |
| 47,00 | | 47,00 | | 41,00 | | 37,00 | | intervall |
| 4,26 | | 4,26 | | 4,88 | | 5,41 | | pixl./int. (1) |
| 50,00 | | 50,00 | | 50,00 | | 50,00 | | endret |
| | | | | | | | | |
| 442,50 | | 48,50 | | 45,50 | | 43,50 | | fra |
| 392,50 | | -1,50 | | -4,50 | | -6,50 | | til |
| | | | | | | | | |
| 4 | | 4 | | 4 | | 4 | | pixl./int. (2) |

**Tabell 3 - Verdier og omregning for ant. brekkasjefeil, kantfeil, sagmerkefeil og misplasserte**

| Antall brekkasje feil | y-verdi | Antall kantfeil | y-verdi | Antall sagmerkefeil | y-verdi | Antall misplasserte | y-verdi | Produkt nr. |
|---|---|---|---|---|---|---|---|---|
| 4,00 | 665,00 | 0,00 | 740,00 | 0,00 | 720,00 | 0,00 | 695,00 | 2 |
| 0,00 | 745,00 | 11,00 | 630,00 | 0,00 | 720,00 | 0,00 | 695,00 | 3 |
| 7,00 | 605,00 | 19,00 | 550,00 | 0,00 | 720,00 | 0,00 | 695,00 | 4 |
| 8,00 | 585,00 | 1,00 | 730,00 | 0,00 | 720,00 | 0,00 | 695,00 | 5 |
| 2,00 | 705,00 | 19,00 | 550,00 | 0,00 | 720,00 | 0,00 | 695,00 | 6 |
| 0,00 | 745,00 | 7,00 | 670,00 | 0,00 | 720,00 | 0,00 | 695,00 | 7 |
| 4,00 | 665,00 | 4,00 | 700,00 | 0,00 | 720,00 | 0,00 | 695,00 | 8 |
| 1,00 | 725,00 | 2,00 | 720,00 | 0,00 | 720,00 | 0,00 | 695,00 | 9 |
| 2,00 | 705,00 | 3,00 | 710,00 | 0,00 | 720,00 | 0,00 | 695,00 | 10 |
| 0,00 | 745,00 | 0,00 | 740,00 | 0,00 | 720,00 | 0,00 | 695,00 | 11 |
| 4,00 | 665,00 | 2,00 | 720,00 | 2,00 | 620,00 | 0,00 | 695,00 | 12 |
| 10,00 | 545,00 | 5,00 | 690,00 | 3,00 | 570,00 | 0,00 | 695,00 | 13 |
| 8,00 | 585,00 | 2,00 | 720,00 | 1,00 | 670,00 | 0,00 | 695,00 | 14 |
| 1,00 | 725,00 | 0,00 | 740,00 | 0,00 | 720,00 | 0,00 | 695,00 | 15 |
| 2,00 | 705,00 | 1,00 | 730,00 | 0,00 | 720,00 | 1,00 | 595,00 | 16 |
| | | | | | | | | |
| 10,00 | | 19,00 | | 3,00 | | 1,00 | | størst |
| 0,00 | | 0,00 | | 0,00 | | 0,00 | | minst |
| 53,00 | | 76,00 | | 6,00 | | 1,00 | | totalt |
| | | | | | | | | |
| 3,53 | | 5,07 | | 0,40 | | 0,07 | | gjennomsnitt |
| | | | | | | | | |
| 10,00 | | 19,00 | | 3,00 | | 1,00 | | intervall |
| 20,00 | | 10,53 | | 66,67 | | 200,00 | | pixl./int. (1) |
| | | 20,00 | | 4,00 | | 2,00 | | endret |
| | | | | | | | | |
| 10,00 | | 19,50 | | 3,50 | | 1,50 | | fra |
| 0,00 | | -0,50 | | -0,50 | | -0,50 | | til |
| | | | | | | | | |
| 20 | | 10 | | 50 | | 100,00 | | pixl./int. (2) |

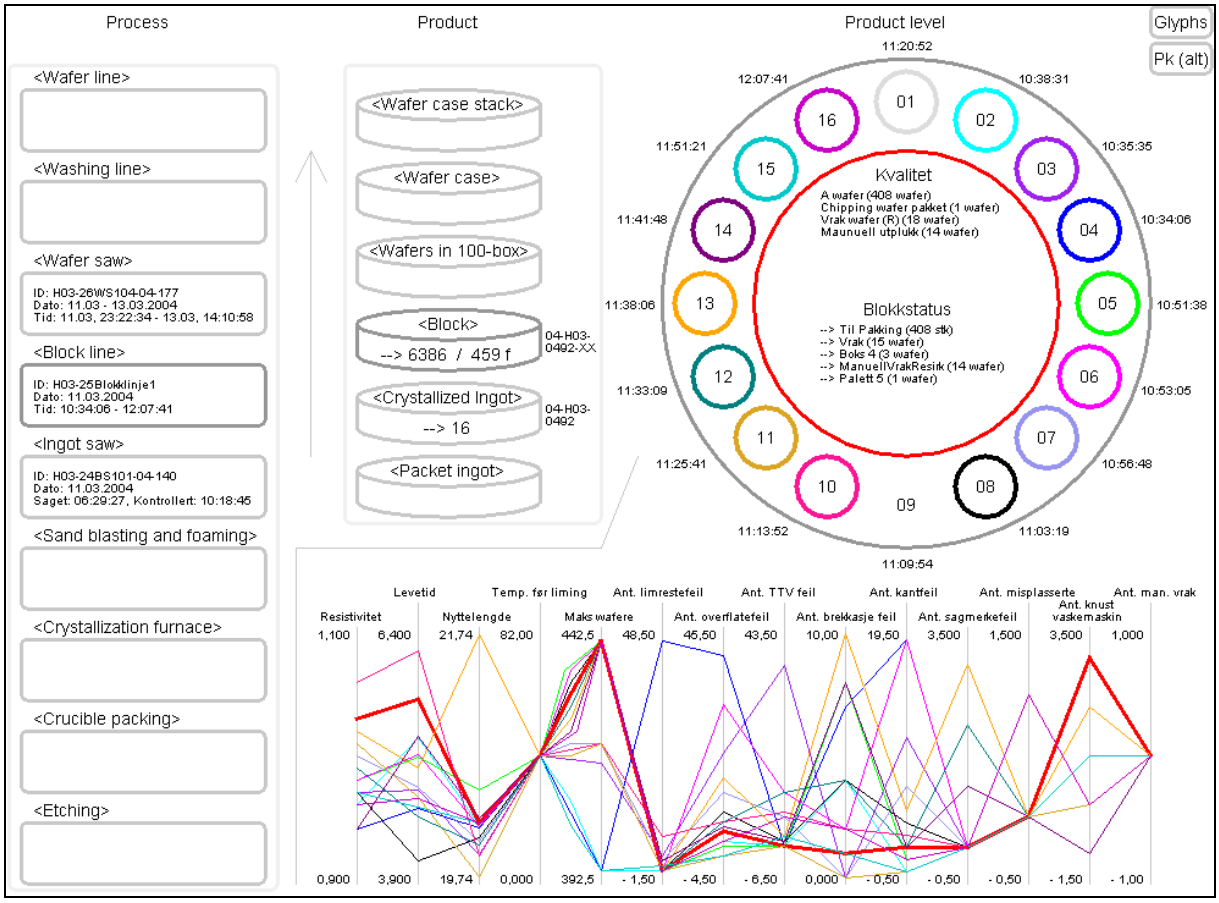
**Tabell 4 - Verdier og omregning for ant. knust i vaskemaskin og manuell vrak**

| Antall knust vaskemaskin | y-verdi | Antall manuell vrak | y-verdi | Produkt nr. |
|---|---|---|---|---|
| 1,00 | 645,00 | 0,00 | 645,00 | 2 |
| 2,00 | 605,00 | 0,00 | 645,00 | 3 |
| 1,00 | 645,00 | 0,00 | 645,00 | 4 |
| 1,00 | 645,00 | 0,00 | 645,00 | 5 |
| 0,00 | 685,00 | 0,00 | 645,00 | 6 |
| 1,00 | 645,00 | 0,00 | 645,00 | 7 |
| 2,00 | 605,00 | 0,00 | 645,00 | 8 |
| 3,00 | 565,00 | 0,00 | 645,00 | 9 |
| -1,00 | 725,00 | 0,00 | 645,00 | 10 |
| 0,00 | 685,00 | 0,00 | 645,00 | 11 |
| 1,00 | 645,00 | 0,00 | 645,00 | 12 |
| 2,00 | 605,00 | 0,00 | 645,00 | 13 |
| -1,00 | 725,00 | 0,00 | 645,00 | 14 |
| 1,00 | 645,00 | 0,00 | 645,00 | 15 |
| 0,00 | 685,00 | 0,00 | 645,00 | 16 |
| | | | | |
| 3,00 | | 0,00 | | størst |
| -1,00 | | 0,00 | | minst |
| 13,00 | | 0,00 | | totalt |
| | | | | |
| 0,87 | | 0,00 | | gjennomsnitt |
| | | | | |
| 4,00 | | 0,00 | | intervall |
| 50,00 | | | | pixl./int. (1) |
| 5,00 | | 2,00 | | endret |
| | | | | |
| 3,50 | | 1,00 | | fra |
| -1,50 | | -1,00 | | til |
| | | | | |
| 40 | | 100,00 | | pixl./int. (2) |

Resten av dataene som er benyttet kan hentes ut fra kildekoden.

# VEDLEGG B: KILDEKODE

Her følger prototypens brukegrensesnittet med utfylte verdier fra en rapport basert på data fra en reell produksjonsprosess (ScanWafer III), og kildekoden.



```
/***
Program: Visualisering, v1
- Hovedprogrammet
- Tegner det grafiske brukergrensesnittet
  og utfører alle interaksjonskommandoene
***/

import java.awt.*;
import java.awt.event.*;
import java.awt.font.*;
import java.awt.geom.*;
import java.awt.image.*;
import java.awt.Toolkit;
import javax.swing.*;

public class Vis_v1
    extends ApplicationFrame
    implements MouseListener, MouseMotionListener {

  // Hovedprosedyre
  public static void main(String[] args) {
    new Vis_v1();
  }

  /*
  ***** Parameterdeklarasjoner *****
  */

  // Glyphs
  protected Image bilde02;
  protected String b02 = "ImageG02.gif";

  protected Image bilde03;
  protected String b03 = "ImageG03.gif";

  protected Image bilde04;
  protected String b04 = "ImageG04.gif";

  protected Image bilde05;
  protected String b05 = "ImageG05.gif";

  protected Image bilde06;
  protected String b06 = "ImageG06.gif";

  protected Image bilde07;
  protected String b07 = "ImageG07.gif";

  protected Image bilde08;
  protected String b08 = "ImageG08.gif";

  protected Image bilde09;
  protected String b09 = "ImageG09.gif";

  protected Image bilde10;
  protected String b10 = "ImageG10.gif";

  protected Image bilde11;
  protected String b11 = "ImageG11.gif";

  protected Image bilde12;
  protected String b12 = "ImageG12.gif";

```





```
  protected Image bilde13;
  protected String b13 = "ImageG13.gif";

  protected Image bilde14;
  protected String b14 = "ImageG14.gif";

  protected Image bilde15;
  protected String b15 = "ImageG15.gif";

  protected Image bilde16;
  protected String b16 = "ImageG16.gif";

  // Glyphs - rammer rundt bildene
  protected RoundRectangle2D R2Dtab02;
  protected RoundRectangle2D R2Dtab03;
  protected RoundRectangle2D R2Dtab04;
  protected RoundRectangle2D R2Dtab05;
  protected RoundRectangle2D R2Dtab06;
  protected RoundRectangle2D R2Dtab07;
  protected RoundRectangle2D R2Dtab08;
  protected RoundRectangle2D R2Dtab09;
  protected RoundRectangle2D R2Dtab10;
  protected RoundRectangle2D R2Dtab11;
  protected RoundRectangle2D R2Dtab12;
  protected RoundRectangle2D R2Dtab13;
  protected RoundRectangle2D R2Dtab14;
  protected RoundRectangle2D R2Dtab15;
  protected RoundRectangle2D R2Dtab16;

  // Prosess
  protected RoundRectangle2D R2D1;
  protected RoundRectangle2D R2D2;
  protected RoundRectangle2D R2D3;
  protected RoundRectangle2D R2D4;
  protected RoundRectangle2D R2D5;
  protected RoundRectangle2D R2D6;
  protected RoundRectangle2D R2D7;
  protected RoundRectangle2D R2D8;
  protected RoundRectangle2D R2D9;

  protected RoundRectangle2D prosessBkg; // rammen rundt;

  // Knapper
  protected RoundRectangle2D R2knapp01;
  protected RoundRectangle2D R2knapp02;

  // Produkt
  protected Arc2D A2D1_1;
  protected Arc2D A2D1_2;
  protected Arc2D A2D2_1;
  protected Arc2D A2D2_2;
  protected Arc2D A2D3_1;
  protected Arc2D A2D3_2;
  protected Arc2D A2D4_1;
  protected Arc2D A2D4_2;
  protected Arc2D A2D5_1;
  protected Arc2D A2D5_2;
  protected Arc2D A2D6_1;
  protected Arc2D A2D6_2;

  protected RoundRectangle2D produktBkg; // rammen rundt

```

```
125   // Produktnivå
126   protected Ellipse2D E;
127   protected Ellipse2D Etmp;
128
129   protected Ellipse2D EP01;
130   protected Ellipse2D EP02;
131   protected Ellipse2D EP03;
132   protected Ellipse2D EP04;
133   protected Ellipse2D EP05;
134   protected Ellipse2D EP06;
135   protected Ellipse2D EP07;
136   protected Ellipse2D EP08;
137   protected Ellipse2D EP09;
138   protected Ellipse2D EP10;
139   protected Ellipse2D EP11;
140   protected Ellipse2D EP12;
141   protected Ellipse2D EP13;
142   protected Ellipse2D EP14;
143   protected Ellipse2D EP15;
144   protected Ellipse2D EP16;
145
146   // Parallellkoodinater
147   protected Line2D PkL = new Line2D.Float();
148
149   protected Point2D[] mP02Points;
150   protected Point2D[] mP03Points;
151   protected Point2D[] mP04Points;
152   protected Point2D[] mP05Points;
153   protected Point2D[] mP06Points;
154   protected Point2D[] mP07Points;
155   protected Point2D[] mP08Points;
156   protected Point2D[] mP09Points;
157   protected Point2D[] mP10Points;
158   protected Point2D[] mP11Points;
159   protected Point2D[] mP12Points;
160   protected Point2D[] mP13Points;
161   protected Point2D[] mP14Points;
162   protected Point2D[] mP15Points;
163   protected Point2D[] mP16Points;
164
165   protected Line2D lPP02 = new Line2D.Float();
166   protected Line2D lPP03 = new Line2D.Float();
167   protected Line2D lPP04 = new Line2D.Float();
168   protected Line2D lPP05 = new Line2D.Float();
169   protected Line2D lPP06 = new Line2D.Float();
170   protected Line2D lPP07 = new Line2D.Float();
171   protected Line2D lPP08 = new Line2D.Float();
172   protected Line2D lPP09 = new Line2D.Float();
173   protected Line2D lPP10 = new Line2D.Float();
174   protected Line2D lPP11 = new Line2D.Float();
175   protected Line2D lPP12 = new Line2D.Float();
176   protected Line2D lPP13 = new Line2D.Float();
177   protected Line2D lPP14 = new Line2D.Float();
178   protected Line2D lPP15 = new Line2D.Float();
179   protected Line2D lPP16 = new Line2D.Float();
180
181   // Pk - knapp 2
182   protected Point2D[] mP02_kn2Points;
183   protected Point2D[] mP03_kn2Points;
184   protected Point2D[] mP04_kn2Points;
185   protected Point2D[] mP05_kn2Points;
186   protected Point2D[] mP06_kn2Points;
```





```
187   protected Point2D[] mP07_kn2Points;
188   protected Point2D[] mP08_kn2Points;
189   protected Point2D[] mP09_kn2Points;
190   protected Point2D[] mP10_kn2Points;
191   protected Point2D[] mP11_kn2Points;
192   protected Point2D[] mP12_kn2Points;
193   protected Point2D[] mP13_kn2Points;
194   protected Point2D[] mP14_kn2Points;
195   protected Point2D[] mP15_kn2Points;
196   protected Point2D[] mP16_kn2Points;
197
198   protected Line2D lPP02_kn2 = new Line2D.Float();
199   protected Line2D lPP03_kn2 = new Line2D.Float();
200   protected Line2D lPP04_kn2 = new Line2D.Float();
201   protected Line2D lPP05_kn2 = new Line2D.Float();
202   protected Line2D lPP06_kn2 = new Line2D.Float();
203   protected Line2D lPP07_kn2 = new Line2D.Float();
204   protected Line2D lPP08_kn2 = new Line2D.Float();
205   protected Line2D lPP09_kn2 = new Line2D.Float();
206   protected Line2D lPP10_kn2 = new Line2D.Float();
207   protected Line2D lPP11_kn2 = new Line2D.Float();
208   protected Line2D lPP12_kn2 = new Line2D.Float();
209   protected Line2D lPP13_kn2 = new Line2D.Float();
210   protected Line2D lPP14_kn2 = new Line2D.Float();
211   protected Line2D lPP15_kn2 = new Line2D.Float();
212   protected Line2D lPP16_kn2 = new Line2D.Float();
213
214   // Variabler (produktnivå, midtstilt sirkel)
215   protected double E1 = 0;
216   protected double E2 = 0;
217   protected double El = 0;
218
219   // Variabler (nullstilling av navigeringsverdier)
220   protected int produkt = 0;
221   protected int produktL = 0;
222   protected int prosess = 0;
223   protected int knappnr = 0;
224
225   protected int avstand = 0;
226
227   // Definerer skrifttyper
228   protected Font skrift = new Font("Lucida Sans Oblique", Font.PLAIN, 14);
229   protected Font skriftPK = new Font("Lucida Sans Oblique", Font.PLAIN, 10);
230
231   // Definerer farger
232   protected Color c1       = new Color(150,150,150);
233   protected Color cProsess = new Color(240,240,240);
234   protected Color cProdukt = new Color(240,240,240);
235   protected Color cStd     = new Color(200,200,200);
236   protected Color cSkrift  = new Color(0,0,0);
237
238
239   protected Color c01 = new Color(220,220,220); // Gray
240   protected Color c02 = new Color(0,255,255); // Cyan
241   protected Color c03 = new Color(160,32,240); // purple
242   protected Color c04 = new Color(0,0,255); // Blue
243   protected Color c05 = new Color(0,255,0); // Green
244   protected Color c06 = new Color(255,0,255); // Magenta
245   protected Color c07 = new Color(151,148,240); // Blueish
246   protected Color c08 = new Color(0,0,0); // Black
247   protected Color c09 = new Color(255,0,0); // Red
248   protected Color c10 = new Color(255,20,147); // Deep pink
```

```
249   protected Color c11 = new Color(218,165,32); // Golden
250   protected Color c12 = new Color(0,127,127); // Deep green
251   protected Color c13 = new Color(255,165,0); // Orange
252   protected Color c14 = new Color(127,0,127); // ??
253   protected Color c15 = new Color(0,200,200); // ??
254   protected Color c16 = new Color(200,0,200); // ??
255
256   // Definerer tykkelser
257   BasicStroke bs0 = new BasicStroke(0);
258   BasicStroke bs1 = new BasicStroke(1);
259   BasicStroke bs2 = new BasicStroke(2);
260   BasicStroke bs3 = new BasicStroke(3);
261   BasicStroke bs4 = new BasicStroke(4);
262   BasicStroke bs5 = new BasicStroke(5);
263
264   // Constructor
265   public Vis_v1() {
266     super("Visualisering v1.0");
267     setSize(1024, 768);
268     center();
269
270     // Lyttere for mushandlinger
271     addMouseListener(this);
272     addMouseMotionListener(this);
273     setVisible(true);
274   }
275
276   // Procedyre for brukergrensesnitt
277   public void paint(Graphics g) {
278     Graphics2D g2 = (Graphics2D)g;
279
280     if (knappnr == 1) {
281
282       bilde02 = Toolkit.getDefaultToolkit().getImage(b02);
283       bilde03 = Toolkit.getDefaultToolkit().getImage(b03);
284       bilde04 = Toolkit.getDefaultToolkit().getImage(b04);
285       bilde05 = Toolkit.getDefaultToolkit().getImage(b05);
286       bilde06 = Toolkit.getDefaultToolkit().getImage(b06);
287       bilde07 = Toolkit.getDefaultToolkit().getImage(b07);
288       bilde08 = Toolkit.getDefaultToolkit().getImage(b08);
289       bilde09 = Toolkit.getDefaultToolkit().getImage(b09);
290       bilde10 = Toolkit.getDefaultToolkit().getImage(b10);
291       bilde11 = Toolkit.getDefaultToolkit().getImage(b11);
292       bilde12 = Toolkit.getDefaultToolkit().getImage(b12);
293       bilde13 = Toolkit.getDefaultToolkit().getImage(b13);
294       bilde14 = Toolkit.getDefaultToolkit().getImage(b14);
295       bilde15 = Toolkit.getDefaultToolkit().getImage(b15);
296       bilde16 = Toolkit.getDefaultToolkit().getImage(b16);
297
298       g2.setStroke(bs3);
299
300       g2.setPaint(c02);
301       R2Dtab02 = new RoundRectangle2D.Double(298,498,100,80,5,5);
302       g2.draw(R2Dtab02);
303       g2.drawImage(bilde02, 300, 500, null);
304
305       g2.setPaint(c03);
306       R2Dtab03 = new RoundRectangle2D.Double(428,498,100,80,5,5);
307       g2.draw(R2Dtab03);
308       g2.drawImage(bilde03, 430, 500, null);
309
310       g2.setPaint(c04);
```





```
311       R2Dtab04 = new RoundRectangle2D.Double(558,498,100,80,5,5);
312       g2.draw(R2Dtab04);
313       g2.drawImage(bilde04, 560, 500, null);
314
315       g2.setPaint(c05);
316       R2Dtab05 = new RoundRectangle2D.Double(688,498,100,80,5,5);
317       g2.draw(R2Dtab05);
318       g2.drawImage(bilde05, 690, 500, null);
319
320       g2.setPaint(c06);
321       R2Dtab06 = new RoundRectangle2D.Double(818,498,100,80,5,5);
322       g2.draw(R2Dtab06);
323       g2.drawImage(bilde06, 820, 500, null);
324
325       g2.setPaint(c07);
326       R2Dtab07 = new RoundRectangle2D.Double(298,588,100,80,5,5);
327       g2.draw(R2Dtab07);
328       g2.drawImage(bilde07, 300, 590, null);
329
330       g2.setPaint(c08);
331       R2Dtab08 = new RoundRectangle2D.Double(428,588,100,80,5,5);
332       g2.draw(R2Dtab08);
333       g2.drawImage(bilde08, 430, 590, null);
334
335       g2.setPaint(c09);
336       R2Dtab09 = new RoundRectangle2D.Double(558,588,100,80,5,5);
337       g2.draw(R2Dtab09);
338       g2.drawImage(bilde09, 560, 590, null);
339
340       g2.setPaint(c10);
341       R2Dtab10 = new RoundRectangle2D.Double(688,588,100,80,5,5);
342       g2.draw(R2Dtab10);
343       g2.drawImage(bilde10, 690, 590, null);
344
345       g2.setPaint(c11);
346       R2Dtab11 = new RoundRectangle2D.Double(818,588,100,80,5,5);
347       g2.draw(R2Dtab11);
348       g2.drawImage(bilde11, 820, 590, null);
349
350       g2.setPaint(c12);
351       R2Dtab12 = new RoundRectangle2D.Double(298,678,100,80,5,5);
352       g2.draw(R2Dtab12);
353       g2.drawImage(bilde12, 300, 680, null);
354
355       g2.setPaint(c13);
356       R2Dtab13 = new RoundRectangle2D.Double(428,678,100,80,5,5);
357       g2.draw(R2Dtab13);
358       g2.drawImage(bilde13, 430, 680, null);
359
360       g2.setPaint(c14);
361       R2Dtab14 = new RoundRectangle2D.Double(558,678,100,80,5,5);
362       g2.draw(R2Dtab14);
363       g2.drawImage(bilde14, 560, 680, null);
364
365       g2.setPaint(c15);
366       R2Dtab15 = new RoundRectangle2D.Double(688,678,100,80,5,5);
367       g2.draw(R2Dtab15);
368       g2.drawImage(bilde15, 690, 680, null);
369
370       g2.setPaint(c16);
371       R2Dtab16 = new RoundRectangle2D.Double(818,678,100,80,5,5);
372       g2.draw(R2Dtab16);
```

```
373       g2.drawImage(bilde16, 820, 680, null);
374
375       g2.setStroke(bs0);
376
377     } // end if knappnr == 1
378
379     /*
380     ***** Tekst *****
381     */
382     g2.setFont(skrift);
383     g2.setPaint(cSkrift);
384
385     g2.drawString("Process", 95, 50);
386     g2.drawString("Product", 350, 50);
387     g2.drawString("Product level", 700, 50);
388
389     // Produkt
390     g2.setFont(skrift);
391     g2.drawString("<Packet ingot>", 327, 417);
392     g2.drawString("<Crystallized Ingot>", 313, 357);
393     g2.drawString("<Block>", 350, 297);
394     g2.drawString("<Wafers in 100-box>", 310, 237);
395     g2.drawString("<Wafer case>", 330, 177);
396     g2.drawString("<Wafer case stack>", 312, 117);
397
398     g2.setFont(skrift);
399     g2.drawString("--> 6386  /  459 f", 320, 322); // Block
400     g2.setFont(skriftPK);
401     g2.drawString("04-H03-", 455, 305); // Block ID
402     g2.drawString("0492-XX", 455, 315); // Block ID
403
404     g2.setFont(skrift);
405     g2.drawString("--> 16", 355, 382); // CIngot (ant. bl.)
406     g2.setFont(skriftPK);
407     g2.drawString("04-H03-", 455, 365); // CIngot ID
408     g2.drawString("0492", 455, 375); // CIngot ID
409
410
411     // Prosess
412     g2.setFont(skrift);
413     g2.drawString("<Etching>", 35, 695);
414     g2.drawString("<Crucible packing>", 35, 620);
415     g2.drawString("<Crystallization furnace>", 35, 545);
416     g2.drawString("<Sand blasting and foaming>", 35, 470);
417     g2.drawString("<Ingot saw>", 35, 395);
418     g2.drawString("<Block line>", 35, 320);
419     g2.drawString("<Wafer saw>", 35, 245);
420     g2.drawString("<Washing line>", 35, 170);
421     g2.drawString("<Wafer line>", 35, 95);
422
423     g2.setFont(skriftPK);
424     g2.drawString("ID: H03-24BS101-04-140", 35, 420); // ISaw
425     g2.drawString("Dato: 11.03.2004", 35, 430); // IDSaw
426     g2.drawString("Saget: 06:29:27, Kontrollert: 10:18:45", 35, 440); // ISaw
427     g2.drawString("ID: H03-25Blokklinje1", 35, 345); // BlL
428     g2.drawString("Dato: 11.03.2004", 35, 355); // BlL
429     g2.drawString("Tid: 10:34:06 - 12:07:41", 35, 365); // BlL
430     g2.drawString("ID: H03-26WS104-04-177", 35, 270); // WSaw
431     g2.drawString("Dato: 11.03 - 13.03.2004", 35, 280); // WSaw
432     g2.drawString("Tid: 11.03, 23:22:34 - 13.03, 14:10:58", 35, 290); // WSaw
433
434     // Knapper
```





```
435     g2.setFont(skrift);
436     g2.drawString("Glyphs", 953, 50);
437     g2.drawString("Pk (alt)", 953, 80);
438
439     if (produktL == 2) {
440       g2.setFont(skrift);
441       g2.drawString("Kvalitet", 725, 175);
442       g2.drawString("Ingot status", 715, 225);
443       g2.drawString("Data", 730, 275);
444
445       g2.setFont(skriftPK);
446       //---
447       g2.drawString("OK (1 stk)", 725, 190);
448       //----
449       g2.drawString("--> Saget (1 stk)", 715, 240);
450       //----
451       g2.drawString("Total fylle vekt (g):", 650, 290);
452       g2.drawString("265 000", 800, 290);
453       g2.drawString("Ønsket resistivitet (ohm/cm):", 650, 300);
454       g2.drawString("1 200", 800, 300);
455       g2.drawString("Beregnet resistivitet (ohm/cm):", 650, 310);
456       g2.drawString("1 200", 800, 310);
457       g2.drawString("Master Alloy behov (g):", 650, 320);
458       g2.drawString("159,87", 800, 320);
459       g2.drawString("Tilsatt Master Alloy (g):", 650, 330);
460       g2.drawString("159,90", 800, 330);
461
462     }
463
464     if (produktL == 3 || prosess == 5 || knappnr > 0) {
465
466       // Produktnivå
467       g2.setFont(skrift);
468       g2.drawString("01", 742, 115);
469       g2.drawString("02", 807, 130);
470       g2.drawString("03", 857, 170);
471       g2.drawString("04", 892, 220);
472       g2.drawString("05", 907, 280);
473       g2.drawString("06", 892, 340);
474       g2.drawString("07", 857, 390);
475       g2.drawString("08", 807, 430);
476       g2.drawString("09", 742, 445);
477       g2.drawString("10", 677, 430);
478       g2.drawString("11", 627, 390);
479       g2.drawString("12", 592, 340);
480       g2.drawString("13", 577, 280);
481       g2.drawString("14", 592, 220);
482       g2.drawString("15", 627, 170);
483       g2.drawString("16", 677, 130);
484
485       g2.setStroke(bs4);
486
487       //Produktnivå
488       //01
489       if (produkt == 1 && prosess != 5)
490           g2.setPaint(Color.white);
491         else
492           g2.setPaint(c01);
493       EP01 = new Ellipse2D.Double(725,85,50,50);
494       g2.draw(EP01);
495
496       //02
```

```
497       if (prosess == 5 || produkt == 2){
498         if (produkt == 2 && prosess != 5)
499           g2.setPaint(Color.white);
500         else
501           g2.setPaint(c01);
502       }
503       else
504         g2.setPaint(c02);
505       EP02 = new Ellipse2D.Double(790,100,50,50);
506       g2.draw(EP02);
507
508       //03
509       if (prosess == 5 || produkt == 3){
510         if (produkt == 3 && prosess != 5)
511           g2.setPaint(Color.white);
512         else
513           g2.setPaint(c01);
514       }
515       else
516         g2.setPaint(c03);
517       EP03 = new Ellipse2D.Double(840,140,50,50);
518       g2.draw(EP03);
519
520       //04
521       if (prosess == 5 || produkt == 4){
522         if (produkt == 4 && prosess != 5)
523           g2.setPaint(Color.white);
524         else
525           g2.setPaint(c01);
526       }
527       else
528         g2.setPaint(c04);
529       EP04 = new Ellipse2D.Double(875,190,50,50);
530       g2.draw(EP04);
531
532       //05
533       if (prosess == 5 || produkt == 5){
534         if (produkt == 5 && prosess != 5)
535           g2.setPaint(Color.white);
536         else
537           g2.setPaint(c01);
538       }
539       else
540         g2.setPaint(c05);
541       EP05 = new Ellipse2D.Double(890,250,50,50);
542       g2.draw(EP05);
543
544       //06
545       if (prosess == 5 || produkt == 6){
546         if (produkt == 6 && prosess != 5)
547           g2.setPaint(Color.white);
548         else
549           g2.setPaint(c01);
550       }
551       else
552         g2.setPaint(c06);
553       EP06 = new Ellipse2D.Double(875,310,50,50);
554       g2.draw(EP06);
555
556       //07
557       if (prosess == 5 || produkt == 7){
558         if (produkt == 7 && prosess != 5)
```





```
559           g2.setPaint(Color.white);
560         else
561           g2.setPaint(c01);
562       }
563       else
564         g2.setPaint(c07);
565       EP07 = new Ellipse2D.Double(840,360,50,50);
566       g2.draw(EP07);
567
568       //08
569       if (prosess == 5 || produkt == 8){
570         if (produkt == 8 && prosess != 5)
571           g2.setPaint(Color.white);
572         else
573           g2.setPaint(c01);
574       }
575       else
576         g2.setPaint(c08);
577       EP08 = new Ellipse2D.Double(790,400,50,50);
578       g2.draw(EP08);
579
580       //09
581       if (prosess == 5 || produkt == 9){
582         if (produkt == 9 && prosess != 5)
583           g2.setPaint(Color.white);
584         else
585           g2.setPaint(c01);
586       }
587       else
588         g2.setPaint(c09);
589       EP09 = new Ellipse2D.Double(725,415,50,50);
590       g2.draw(EP09);
591
592       //10
593       if (prosess == 5 || produkt == 10){
594         if (produkt == 10 && prosess != 5)
595           g2.setPaint(Color.white);
596         else
597           g2.setPaint(c01);
598       }
599       else
600         g2.setPaint(c10);
601       EP10 = new Ellipse2D.Double(660,400,50,50);
602       g2.draw(EP10);
603
604       //11
605       if (prosess == 5 || produkt == 11){
606         if (produkt == 11 && prosess != 5)
607           g2.setPaint(Color.white);
608         else
609           g2.setPaint(c01);
610       }
611       else
612         g2.setPaint(c11);
613       EP11 = new Ellipse2D.Double(610,360,50,50);
614       g2.draw(EP11);
615
616       //12
617       if (prosess == 5 || produkt == 12){
618         if (produkt == 12 && prosess != 5)
619           g2.setPaint(Color.white);
620         else
```

```
621             g2.setPaint(c01);
622         }
623         else
624           g2.setPaint(c12);
625         EP12 = new Ellipse2D.Double(575,310,50,50);
626         g2.draw(EP12);
627
628         //13
629         if (prosess == 5 || produkt == 13){
630           if (produkt == 13 && prosess != 5)
631             g2.setPaint(Color.white);
632           else
633             g2.setPaint(c01);
634         }
635         else
636           g2.setPaint(c13);
637         EP13 = new Ellipse2D.Double(560,250,50,50);
638         g2.draw(EP13);
639
640         //14
641         if (prosess == 5 || produkt == 14){
642           if (produkt == 14 && prosess != 5)
643             g2.setPaint(Color.white);
644           else
645             g2.setPaint(c01);
646         }
647         else
648           g2.setPaint(c14);
649         EP14 = new Ellipse2D.Double(575,190,50,50);
650         g2.draw(EP14);
651
652         //15
653         if (prosess == 5 || produkt == 15){
654           if (produkt == 15 && prosess != 5)
655             g2.setPaint(Color.white);
656           else
657             g2.setPaint(c01);
658         }
659         else
660           g2.setPaint(c15);
661         EP15 = new Ellipse2D.Double(610,140,50,50);
662         g2.draw(EP15);
663
664         //16
665         if (prosess == 5 || produkt == 16){
666           if (produkt == 16 && prosess != 5)
667             g2.setPaint(Color.white);
668           else
669             g2.setPaint(c01);
670         }
671         else
672           g2.setPaint(c16); //16
673         EP16 = new Ellipse2D.Double(660,100,50,50);
674         g2.draw(EP16);
675
676       }
677
678       if (produktL == 3 || knappnr > 0) {
679
680         if (produkt == 1) {
681           g2.setPaint(cSkrift);
682           g2.setFont(skrift);
```





```
683           g2.drawString("Kvalitet", 725, 175);
684           g2.drawString("Blokkstatus", 715, 280);
685
686           g2.setFont(skriftPK);
687           g2.drawString("OK (1 stk)", 680, 190);
688           //----
689           g2.drawString("I blokklinje (1 stk)", 680, 300);
690         }
691
692         if (produkt == 2) {
693           g2.setPaint(cSkrift);
694           g2.setFont(skrift);
695           g2.drawString("Kvalitet", 725, 175);
696           g2.drawString("Blokkstatus", 715, 285);
697
698           g2.setFont(skriftPK);
699           g2.drawString("A wafer (368 wafer)", 680, 190);
700           g2.drawString("Vrak wafer (R) (15 wafer)", 680, 200);
701           g2.drawString("Diverse wafer (R) (2 wafer)", 680, 210);
702           g2.drawString("Maunuell utplukk (9 wafer)", 680, 220);
703           //----
704           g2.drawString("--> Til pakking (368 stk)", 680, 300);
705           g2.drawString("--> Boks 5 (2 wafer)", 680, 310);
706           g2.drawString("--> Vrak (15 wafer)", 680, 320);
707           g2.drawString("--> ManuellVrakResirk (9 wafer)", 680, 330);
708         }
709
710         if (produkt == 3) {
711           g2.setPaint(cSkrift);
712           g2.setFont(skrift);
713           g2.drawString("Kvalitet", 725, 175);
714           g2.drawString("Blokkstatus", 715, 285);
715
716           g2.setFont(skriftPK);
717           g2.drawString("A wafer (4 wafer)", 680, 190);
718           g2.drawString("Chipping wafer pakket (24 wafer)", 680, 200);
719           g2.drawString("Vrak wafer (R) (19 wafer)", 680, 210);
720           g2.drawString("Resirk wafer (R) (333 wafer)", 680, 220);
721           g2.drawString("TTV wafer pakket (36 wafer)", 680, 230);
722           //----
723           g2.drawString("--> Til Pakking (4 stk)", 680, 300);
724           g2.drawString("--> Vrak (13 wafer)", 680, 310);
725           g2.drawString("--> Boks 4 (6 wafer)", 680, 320);
726           g2.drawString("--> Boks 3 (333 wafer)", 680, 330);
727           g2.drawString("--> Palett 5 (24 wafer)", 680, 340);
728           g2.drawString("--> Palett 3 (36 wafer)", 680, 350);
729         }
730
731         if (produkt == 4) {
732           g2.setPaint(cSkrift);
733           g2.setFont(skrift);
734           g2.drawString("Kvalitet", 725, 175);
735           g2.drawString("Blokkstatus", 715, 285);
736
737           g2.setFont(skriftPK);
738           g2.drawString("A wafer (292 wafer)", 680, 190);
739           g2.drawString("Chipping wafer pakket (2 wafer)", 680, 200);
740           g2.drawString("Vrak wafer (R) (70 wafer)", 680, 210);
741           g2.drawString("Maunuell utplukk (29 wafer)", 680, 220);
742           g2.drawString("Diverse wafer (R) (1 wafer)", 680, 230);
743           //----
744           g2.drawString("--> Palett 5 (2 wafer)", 680, 300);
```

```
745         g2.drawString("--> Til pakking (292 stk)", 680, 310);
746         g2.drawString("--> Vrak (70 wafer)", 680, 320);
747         g2.drawString("--> Boks 5 (1 wafer)", 680, 330);
748         g2.drawString("--> ManuellVrakResirk (29 wafer)", 680, 340);
749       }
750
751       if (produkt == 5) {
752         g2.setPaint(cSkrift);
753         g2.setFont(skrift);
754         g2.drawString("Kvalitet", 725, 175);
755         g2.drawString("Blokkstatus", 715, 285);
756
757         g2.setFont(skriftPK);
758         g2.drawString("A wafer (398 wafer)", 680, 190);
759         g2.drawString("Chipping wafer pakket (7 wafer)", 680, 200);
760         g2.drawString("Vrak wafer (R) (32 wafer)", 680, 210);
761         g2.drawString("Maunuell utplukk (4 wafer)", 680, 220);
762         //----
763         g2.drawString("--> Til Pakking (398 stk)", 680, 300);
764         g2.drawString("--> Vrak (26 wafer)", 680, 310);
765         g2.drawString("--> Boks 4 (6 wafer)", 680, 320);
766         g2.drawString("--> ManuellVrakResirk (4 wafer)", 680, 330);
767         g2.drawString("--> Palett 5 (7 wafer)", 680, 340);
768       }
769
770       if (produkt == 6) {
771         g2.setPaint(cSkrift);
772         g2.setFont(skrift);
773         g2.drawString("Kvalitet", 725, 175);
774         g2.drawString("Blokkstatus", 715, 285);
775
776         g2.setFont(skriftPK);
777         g2.drawString("A wafer (393 wafer)", 680, 190);
778         g2.drawString("Chipping wafer pakket (5 wafer)", 680, 200);
779         g2.drawString("Vrak wafer (R) (14 wafer)", 680, 210);
780         g2.drawString("Maunuell utplukk (20 wafer)", 680, 220);
781         g2.drawString("Resirk wafer (R) (1 wafer)", 680, 230);
782         g2.drawString("TTV wafer pakket (8 wafer)", 680, 240);
783         //----
784         g2.drawString("--> Til Pakking (393 stk)", 680, 300);
785         g2.drawString("--> Vrak (12 wafer)", 680, 310);
786         g2.drawString("--> Boks 4 (2 wafer)", 680, 320);
787         g2.drawString("--> Boks 3 (1 wafer)", 680, 330);
788         g2.drawString("--> Palett 5 (5 wafer)", 680, 340);
789         g2.drawString("--> ManuellVrakResirk (20 wafer)", 680, 350);
790         g2.drawString("--> Palett 3 (8 wafer)", 680, 360);
791       }
792
793       if (produkt == 7) {
794         g2.setPaint(cSkrift);
795         g2.setFont(skrift);
796         g2.drawString("Kvalitet", 725, 175);
797         g2.drawString("Blokkstatus", 715, 285);
798
799         g2.setFont(skriftPK);
800         g2.drawString("A wafer (373 wafer)", 680, 190);
801         g2.drawString("Chipping wafer pakket (3 wafer)", 680, 200);
802         g2.drawString("Vrak wafer (R) (17 wafer)", 680, 210);
803         g2.drawString("Resirk wafer (R) (3 wafer)", 680, 220);
804         g2.drawString("Maunuell utplukk (21 wafer)", 680, 230);
805         g2.drawString("TTV wafer pakket (3 wafer)", 680, 240);
806         //----
```





```
807         g2.drawString("--> Til Pakking (373 stk)", 680, 300);
808         g2.drawString("--> Boks 3 (3 wafer)", 680, 310);
809         g2.drawString("--> Boks 4 (1 wafer)", 680, 320);
810         g2.drawString("--> Vrak (16 wafer)", 680, 330);
811         g2.drawString("--> ManuellVrakResirk (21 wafer)", 680, 340);
812         g2.drawString("--> Palett 5 (3 wafer)", 680, 350);
813         g2.drawString("--> Palett 3 (3 wafer)", 680, 360);
814       }
815
816       if (produkt == 8) {
817         g2.setPaint(cSkrift);
818         g2.setFont(skrift);
819         g2.drawString("Kvalitet", 725, 175);
820         g2.drawString("Blokkstatus", 715, 285);
821
822         g2.setFont(skriftPK);
823         g2.drawString("A wafer (405 wafer)", 680, 190);
824         g2.drawString("Chipping wafer pakket (4 wafer)", 680, 200);
825         g2.drawString("Vrak wafer (R) (24 wafer)", 680, 210);
826         g2.drawString("Maunuell utplukk (8 wafer)", 680, 220);
827         //----
828         g2.drawString("--> Til Pakking (405 stk)", 680, 300);
829         g2.drawString("--> Vrak (20 wafer)", 680, 310);
830         g2.drawString("--> Boks 4 (4 wafer)", 680, 320);
831         g2.drawString("--> Palett 5 (4 wafer)", 680, 330);
832         g2.drawString("--> ManuellVrakResirk (8 wafer)", 680, 340);
833       }
834
835       if (produkt == 9) {
836         g2.setPaint(cSkrift);
837         g2.setFont(skrift);
838         g2.drawString("Kvalitet", 725, 175);
839         g2.drawString("Blokkstatus", 715, 285);
840
841         g2.setFont(skriftPK);
842         g2.drawString("A wafer (408 wafer)", 680, 190);
843         g2.drawString("Chipping wafer pakket (1 wafer)", 680, 200);
844         g2.drawString("Vrak wafer (R) (18 wafer)", 680, 210);
845         g2.drawString("Maunuell utplukk (14 wafer)", 680, 220);
846         //----
847         g2.drawString("--> Til Pakking (408 stk)", 680, 300);
848         g2.drawString("--> Vrak (15 wafer)", 680, 310);
849         g2.drawString("--> Boks 4 (3 wafer)", 680, 320);
850         g2.drawString("--> ManuellVrakResirk (14 wafer)", 680, 330);
851         g2.drawString("--> Palett 5 (1 wafer)", 680, 340);
852       }
853
854       if (produkt == 10) {
855         g2.setPaint(cSkrift);
856         g2.setFont(skrift);
857         g2.drawString("Kvalitet", 725, 175);
858         g2.drawString("Blokkstatus", 715, 285);
859
860         g2.setFont(skriftPK);
861         g2.drawString("A wafer (378 wafer)", 680, 190);
862         g2.drawString("Chipping wafer pakket (14 wafer)", 680, 200);
863         g2.drawString("Vrak wafer (R) (21 wafer)", 680, 210);
864         g2.drawString("Resirk wafer (R) (2 wafer)", 680, 220);
865         g2.drawString("TTV wafer pakket (5 wafer)", 680, 230);
866         //----
867         g2.drawString("--> Til Pakking (378 stk)", 680, 300);
868         g2.drawString("--> Vrak (20 wafer)", 680, 310);
```

```
869           g2.drawString("--> Boks 4 (1 wafer)", 680, 320);
870           g2.drawString("--> Boks 3 (2 wafer)", 680, 330);
871           g2.drawString("--> Palett 5 (14 wafer)", 680, 340);
872           g2.drawString("--> Palett 3 (5 wafer)", 680, 350);
873         }
874
875         if (produkt == 11) {
876           g2.setPaint(cSkrift);
877           g2.setFont(skrift);
878           g2.drawString("Kvalitet", 725, 175);
879           g2.drawString("Blokkstatus", 715, 285);
880
881           g2.setFont(skriftPK);
882           g2.drawString("A wafer (234 wafer)", 680, 190);
883           g2.drawString("Maunuell utplukk (186 wafer)", 680, 200);
884           //----
885           g2.drawString("--> Til Pakking (234 stk)", 680, 300);
886           g2.drawString("--> ManuellVrakResirk (186 wafer)", 680, 310);
887         }
888
889         if (produkt == 12) {
890           g2.setPaint(cSkrift);
891           g2.setFont(skrift);
892           g2.drawString("Kvalitet", 725, 175);
893           g2.drawString("Blokkstatus", 715, 285);
894
895           g2.setFont(skriftPK);
896           g2.drawString("A wafer (383 wafer)", 680, 190);
897           g2.drawString("Chipping wafer pakket (8 wafer)", 680, 200);
898           g2.drawString("Vrak wafer (R) (23 wafer)", 680, 210);
899           g2.drawString("Maunuell utplukk (14 wafer)", 680, 220);
900           g2.drawString("Resirk wafer (R) (2 wafer)", 680, 230);
901           g2.drawString("TTV wafer pakket (9 wafer)", 680, 240);
902           g2.drawString("Sagmerke wafer pakket (2 wafer)", 680, 250);
903           //----
904           g2.drawString("--> Til Pakking (383 stk)", 680, 300);
905           g2.drawString("--> Boks 3 (2 wafer)", 680, 310);
906           g2.drawString("--> Boks 4 (2 wafer)", 680, 320);
907           g2.drawString("--> Vrak (21 wafer)", 680, 330);
908           g2.drawString("--> ManuellVrakResirk (14 wafer)", 680, 340);
909           g2.drawString("--> Palett 5 (8 wafer)", 680, 350);
910           g2.drawString("--> Palett 4 (2 wafer)", 680, 360);
911           g2.drawString("--> Palett 3 (9 wafer)", 680, 370);
912         }
913
914         if (produkt == 13) {
915           g2.setPaint(cSkrift);
916           g2.setFont(skrift);
917           g2.drawString("Kvalitet", 725, 175);
918           g2.drawString("Blokkstatus", 715, 285);
919
920           g2.setFont(skriftPK);
921           g2.drawString("A wafer (289 wafer)", 680, 190);
922           g2.drawString("Chipping wafer pakket (3 wafer)", 680, 200);
923           g2.drawString("Vrak wafer (R) (36 wafer)", 680, 210);
924           g2.drawString("Resirk wafer (R) (1 wafer)", 680, 220);
925           g2.drawString("Maunuell utplukk (112 wafer)", 680, 230);
926           //----
927           g2.drawString("--> Til Pakking (289 stk)", 680, 300);
928           g2.drawString("--> Boks 3 (1 wafer)", 680, 310);
929           g2.drawString("--> Vrak (36 wafer)", 680, 320);
930           g2.drawString("--> Palett 5 (3 wafer)", 680, 330);
```





```
931           g2.drawString("--> ManuellVrakResirk (112 wafer)", 680, 340);
932         }
933
934         if (produkt == 14) {
935           g2.setPaint(cSkrift);
936           g2.setFont(skrift);
937           g2.drawString("Kvalitet", 725, 175);
938           g2.drawString("Blokkstatus", 715, 285);
939
940           g2.setFont(skriftPK);
941           g2.drawString("A wafer (20 wafer)", 680, 190);
942           g2.drawString("Chipping wafer pakket (2 wafer)", 680, 200);
943           g2.drawString("Vrak wafer (R) (30 wafer)", 680, 210);
944           g2.drawString("Resirk wafer (R) (389 wafer)", 680, 220);
945           //----
946           g2.drawString("--> Til Pakking (20 stk)", 680, 300);
947           g2.drawString("--> Boks 3 (389 wafer)", 680, 310);
948           g2.drawString("--> Vrak (19 wafer)", 680, 320);
949           g2.drawString("--> Boks 4 (11 wafer)", 680, 330);
950           g2.drawString("--> Palett 5 (2 wafer)", 680, 340);
951         }
952
953         if (produkt == 15) {
954           g2.setPaint(cSkrift);
955           g2.setFont(skrift);
956           g2.drawString("Kvalitet", 725, 175);
957           g2.drawString("Blokkstatus", 715, 285);
958
959           g2.setFont(skriftPK);
960           g2.drawString("A wafer (362 wafer)", 680, 190);
961           g2.drawString("Vrak wafer (R) (18 wafer)", 680, 200);
962           g2.drawString("Maunuell utplukk (12 wafer)", 680, 210);
963           g2.drawString("TTV wafer pakket (2 wafer)", 680, 220);
964           //----
965           g2.drawString("--> Palett 3 (2 wafer)", 680, 300);
966           g2.drawString("--> Til pakking (362 stk)", 680, 310);
967           g2.drawString("--> Vrak (17 wafer)", 680, 320);
968           g2.drawString("--> Boks 4 (1 wafer)", 680, 330);
969           g2.drawString("--> ManuellVrakResirk (12 wafer)", 680, 340);
970         }
971
972         if (produkt == 16) {
973           g2.setPaint(cSkrift);
974           g2.setFont(skrift);
975           g2.drawString("Kvalitet", 725, 175);
976           g2.drawString("Blokkstatus", 715, 285);
977
978           g2.setFont(skriftPK);
979           g2.drawString("A wafer (411 wafer)", 680, 190);
980           g2.drawString("Chipping wafer pakket (2 wafer)", 680, 200);
981           g2.drawString("Vrak wafer (R) (8 wafer)", 680, 210);
982           g2.drawString("Maunuell utplukk (15 wafer)", 680, 220);
983           g2.drawString("TTV wafer pakket (5 wafer)", 680, 230);
984           //----
985           g2.drawString("--> Til Pakking (411 stk)", 680, 300);
986           g2.drawString("--> Vrak (7 wafer)", 680, 310);
987           g2.drawString("--> Boks 4 (1 wafer)", 680, 320);
988           g2.drawString("--> ManuellVrakResirk (15 wafer)", 680, 330);
989           g2.drawString("--> Palett 5 (2 wafer)", 680, 340);
990           g2.drawString("--> Palett 3 (5 wafer)", 680, 350);
991         }
992
```

```
993         g2.setPaint(cSkrift);
994         g2.setFont(skriftPK);
995         g2.drawString("11:20:52", 730,  68); // 01
996         g2.drawString("10:38:31", 842,  95); // 02
997         g2.drawString("10:35:35", 910, 150); // 03
998         g2.drawString("10:34:06", 942, 210); // 04
999         g2.drawString("10:51:38", 955, 280); // 05
1000        g2.drawString("10:53:05", 942, 350); // 06
1001        g2.drawString("10:56:48", 910, 410); // 07
1002        g2.drawString("11:03:19", 842, 465); // 08
1003        g2.drawString("11:09:54", 730, 492); // 09
1004        g2.drawString("11:13:52", 612, 465); // 10
1005        g2.drawString("11:25:41", 545, 410); // 11
1006        g2.drawString("11:33:09", 513, 350); // 12
1007        g2.drawString("11:38:06", 503, 280); // 13
1008        g2.drawString("11:41:48", 513, 210); // 14
1009        g2.drawString("11:51:21", 545, 150); // 15
1010        g2.drawString("12:07:41", 612,  95); // 16
1011
1012      }
1013
1014      if (produktL == 3 || knappnr == 2){
1015
1016        // Parallellkoordinater (akseverdier og -navn)
1017        g2.setPaint(cSkrift);
1018        g2.setFont(skriftPK);
1019        g2.drawString("Resistivitet", 270, 535);
1020        g2.drawString("Levetid", 330, 515);
1021        g2.drawString("Nyttelengde", 370, 535);
1022        g2.drawString("Temp. før liming", 410, 515);
1023        g2.drawString("Maks wafere", 470, 535);
1024        g2.drawString("Ant. limrestefeil", 510, 515);
1025        g2.drawString("Ant. overflatefeil", 560, 535);
1026        g2.drawString("Ant. TTV feil", 615, 515);
1027        g2.drawString("Ant. brekkasje feil", 660, 535);
1028        g2.drawString("Ant. kantfeil", 720, 515);
1029        g2.drawString("Ant. sagmerkefeil", 760, 535);
1030        g2.drawString("Ant. misplasserte", 810, 515);
1031        g2.drawString(("Ant. knust"), 875, 525);
1032        g2.drawString(("vaskemaskin"), 870, 535);
1033        g2.drawString("Ant. man. vrak", 920, 515);
1034
1035        g2.drawString("1,100", 268, 550);
1036        g2.drawString("6,400", 318, 550);
1037        g2.drawString("21,74", 368, 550);
1038        g2.drawString("82,00", 418, 550);
1039        g2.drawString("442,5", 468, 550);
1040        g2.drawString("48,50", 518, 550);
1041        g2.drawString("45,50", 568, 550);
1042        g2.drawString("43,50", 618, 550);
1043        g2.drawString("10,00", 668, 550);
1044        g2.drawString("19,50", 718, 550);
1045        g2.drawString("3,500", 768, 550);
1046        g2.drawString("1,500", 818, 550);
1047        g2.drawString("3,500", 868, 550);
1048        g2.drawString("1,000", 918, 550);
1049
1050        g2.drawString("0,900", 268, 750);
1051        g2.drawString("3,900", 318, 750);
1052        g2.drawString("19,74", 368, 750);
1053        g2.drawString("0,000", 418, 750);
1054        g2.drawString("392,5", 468, 750);
```





```
1055        g2.drawString("- 1,50", 518, 750);
1056        g2.drawString("- 4,50", 568, 750);
1057        g2.drawString("- 6,50", 618, 750);
1058        g2.drawString("0,000", 668, 750);
1059        g2.drawString("- 0,50", 718, 750);
1060        g2.drawString("- 0,50", 768, 750);
1061        g2.drawString("- 0,50", 818, 750);
1062        g2.drawString("- 1,50", 868, 750);
1063        g2.drawString("- 1,00", 918, 750);
1064
1065
1066        // Parallellkoordinater (akser)
1067        g2.setStroke(bs0);
1068        g2.setPaint(cStd);
1069
1070        avstand = 300;
1071        for (int i = 0; i < 7; i++) {
1072          PkL.setLine(avstand,540,avstand,750);
1073          g2.draw(PkL);
1074          avstand = avstand + 50;
1075          PkL.setLine(avstand,520,avstand,750);
1076          g2.draw(PkL);
1077          avstand = avstand + 50;
1078        }
1079
1080      } // end produktL == 3 || knappnr == 2
1081
1082      if (produktL == 3){
1083        // Parallellkoordinater (produktverdier)
1084        // Produkt 02
1085        mP02Points = new Point2D[14];
1086        mP02Points[0]  = new Point2D.Double(300, 685);
1087        mP02Points[1]  = new Point2D.Double(350, 630);
1088        mP02Points[2]  = new Point2D.Double(400, 702);
1089        mP02Points[3]  = new Point2D.Double(450, 645);
1090        mP02Points[4]  = new Point2D.Double(500, 739);
1091        mP02Points[5]  = new Point2D.Double(550, 739);
1092        mP02Points[6]  = new Point2D.Double(600, 715);
1093        mP02Points[7]  = new Point2D.Double(650, 719);
1094        mP02Points[8]  = new Point2D.Double(700, 665);
1095        mP02Points[9]  = new Point2D.Double(750, 740);
1096        mP02Points[10] = new Point2D.Double(800, 720);
1097        mP02Points[11] = new Point2D.Double(850, 695);
1098        mP02Points[12] = new Point2D.Double(900, 645);
1099        mP02Points[13] = new Point2D.Double(950, 645);
1100
1101        g2.setPaint(c02);
1102
1103        if (produkt == 2)
1104          g2.setStroke(bs3);
1105        else
1106          g2.setStroke(bs1);
1107
1108        for (int i = 0; i < mP02Points.length - 1; i++) {
1109          lPP02.setLine(mP02Points[i], mP02Points[i+1]);
1110          g2.draw(lPP02);
1111        }
1112
1113        // Produkt 03
1114        mP03Points = new Point2D[14];
1115        mP03Points[0]  = new Point2D.Double(300, 675);
1116        mP03Points[1]  = new Point2D.Double(350, 673);
```

```
1117        mP03Points[2]  = new Point2D.Double(400, 726);
1118        mP03Points[3]  = new Point2D.Double(450, 645);
1119        mP03Points[4]  = new Point2D.Double(500, 651);
1120        mP03Points[5]  = new Point2D.Double(550, 727);
1121        mP03Points[6]  = new Point2D.Double(600, 643);
1122        mP03Points[7]  = new Point2D.Double(650, 571);
1123        mP03Points[8]  = new Point2D.Double(700, 745);
1124        mP03Points[9]  = new Point2D.Double(750, 630);
1125        mP03Points[10] = new Point2D.Double(800, 720);
1126        mP03Points[11] = new Point2D.Double(850, 695);
1127        mP03Points[12] = new Point2D.Double(900, 605);
1128        mP03Points[13] = new Point2D.Double(950, 645);
1129
1130        g2.setPaint(c03);
1131
1132        if (produkt == 3)
1133          g2.setStroke(bs3);
1134        else
1135          g2.setStroke(bs1);
1136
1137        for (int i = 0; i < mP03Points.length - 1; i++) {
1138          lPP03.setLine(mP03Points[i], mP03Points[i+1]);
1139          g2.draw(lPP03);
1140        }
1141
1142        // Produkt 04
1143        mP04Points = new Point2D[14];
1144        mP04Points[0]  = new Point2D.Double(300, 705);
1145        mP04Points[1]  = new Point2D.Double(350, 688);
1146        mP04Points[2]  = new Point2D.Double(400, 704);
1147        mP04Points[3]  = new Point2D.Double(450, 645);
1148        mP04Points[4]  = new Point2D.Double(500, 739);
1149        mP04Points[5]  = new Point2D.Double(550, 551);
1150        mP04Points[6]  = new Point2D.Double(600, 563);
1151        mP04Points[7]  = new Point2D.Double(650, 719);
1152        mP04Points[8]  = new Point2D.Double(700, 605);
1153        mP04Points[9]  = new Point2D.Double(750, 550);
1154        mP04Points[10] = new Point2D.Double(800, 720);
1155        mP04Points[11] = new Point2D.Double(850, 695);
1156        mP04Points[12] = new Point2D.Double(900, 645);
1157        mP04Points[13] = new Point2D.Double(950, 645);
1158
1159        g2.setPaint(c04);
1160
1161        if (produkt == 4)
1162          g2.setStroke(bs3);
1163        else
1164          g2.setStroke(bs1);
1165
1166        for (int i = 0; i < mP04Points.length - 1; i++) {
1167          lPP04.setLine(mP04Points[i], mP04Points[i+1]);
1168          g2.draw(lPP04);
1169        }
1170
1171        // Produkt 05
1172        mP05Points = new Point2D[14];
1173        mP05Points[0]  = new Point2D.Double(300, 665);
1174        mP05Points[1]  = new Point2D.Double(350, 646);
1175        mP05Points[2]  = new Point2D.Double(400, 673);
1176        mP05Points[3]  = new Point2D.Double(450, 645);
1177        mP05Points[4]  = new Point2D.Double(500, 551);
1178        mP05Points[5]  = new Point2D.Double(550, 739);
```





```
1179        mP05Points[6]  = new Point2D.Double(600, 719);
1180        mP05Points[7]  = new Point2D.Double(650, 719);
1181        mP05Points[8]  = new Point2D.Double(700, 585);
1182        mP05Points[9]  = new Point2D.Double(750, 730);
1183        mP05Points[10] = new Point2D.Double(800, 720);
1184        mP05Points[11] = new Point2D.Double(850, 695);
1185        mP05Points[12] = new Point2D.Double(900, 645);
1186        mP05Points[13] = new Point2D.Double(950, 645);
1187
1188        g2.setPaint(c05);
1189
1190        if (produkt == 5)
1191          g2.setStroke(bs3);
1192        else
1193          g2.setStroke(bs1);
1194
1195        for (int i = 0; i < mP05Points.length - 1; i++) {
1196          lPP05.setLine(mP05Points[i], mP05Points[i+1]);
1197          g2.draw(lPP05);
1198        }
1199
1200        // Produkt 06
1201        mP06Points = new Point2D[14];
1202        mP06Points[0]  = new Point2D.Double(300, 665);
1203        mP06Points[1]  = new Point2D.Double(350, 644);
1204        mP06Points[2]  = new Point2D.Double(400, 703);
1205        mP06Points[3]  = new Point2D.Double(450, 645);
1206        mP06Points[4]  = new Point2D.Double(500, 551);
1207        mP06Points[5]  = new Point2D.Double(550, 739);
1208        mP06Points[6]  = new Point2D.Double(600, 603);
1209        mP06Points[7]  = new Point2D.Double(650, 675);
1210        mP06Points[8]  = new Point2D.Double(700, 705);
1211        mP06Points[9]  = new Point2D.Double(750, 550);
1212        mP06Points[10] = new Point2D.Double(800, 720);
1213        mP06Points[11] = new Point2D.Double(850, 695);
1214        mP06Points[12] = new Point2D.Double(900, 685);
1215        mP06Points[13] = new Point2D.Double(950, 645);
1216
1217        g2.setPaint(c06);
1218
1219        if (produkt == 6)
1220          g2.setStroke(bs3);
1221        else
1222          g2.setStroke(bs1);
1223
1224        for (int i = 0; i < mP06Points.length - 1; i++) {
1225          lPP06.setLine(mP06Points[i], mP06Points[i+1]);
1226          g2.draw(lPP06);
1227        }
1228
1229        // Produkt 07
1230        mP07Points = new Point2D[14];
1231        mP07Points[0]  = new Point2D.Double(300, 645);
1232        mP07Points[1]  = new Point2D.Double(350, 671);
1233        mP07Points[2]  = new Point2D.Double(400, 715);
1234        mP07Points[3]  = new Point2D.Double(450, 645);
1235        mP07Points[4]  = new Point2D.Double(500, 635);
1236        mP07Points[5]  = new Point2D.Double(550, 731);
1237        mP07Points[6]  = new Point2D.Double(600, 675);
1238        mP07Points[7]  = new Point2D.Double(650, 695);
1239        mP07Points[8]  = new Point2D.Double(700, 745);
1240        mP07Points[9]  = new Point2D.Double(750, 670);
```

```
1241        mP07Points[10] = new Point2D.Double(800, 720);
1242        mP07Points[11] = new Point2D.Double(850, 695);
1243        mP07Points[12] = new Point2D.Double(900, 645);
1244        mP07Points[13] = new Point2D.Double(950, 645);
1245
1246        g2.setPaint(c07);
1247
1248        if (produkt == 7)
1249          g2.setStroke(bs3);
1250        else
1251          g2.setStroke(bs1);
1252
1253        for (int i = 0; i < mP07Points.length - 1; i++) {
1254          lPP07.setLine(mP07Points[i], mP07Points[i+1]);
1255          g2.draw(lPP07);
1256        }
1257
1258        // Produkt 08
1259        mP08Points = new Point2D[14];
1260        mP08Points[0]  = new Point2D.Double(300, 675);
1261        mP08Points[1]  = new Point2D.Double(350, 731);
1262        mP08Points[2]  = new Point2D.Double(400, 712);
1263        mP08Points[3]  = new Point2D.Double(450, 645);
1264        mP08Points[4]  = new Point2D.Double(500, 551);
1265        mP08Points[5]  = new Point2D.Double(550, 739);
1266        mP08Points[6]  = new Point2D.Double(600, 691);
1267        mP08Points[7]  = new Point2D.Double(650, 715);
1268        mP08Points[8]  = new Point2D.Double(700, 665);
1269        mP08Points[9]  = new Point2D.Double(750, 700);
1270        mP08Points[10] = new Point2D.Double(800, 720);
1271        mP08Points[11] = new Point2D.Double(850, 695);
1272        mP08Points[12] = new Point2D.Double(900, 605);
1273        mP08Points[13] = new Point2D.Double(950, 645);
1274
1275        g2.setPaint(c08);
1276
1277        if (produkt == 8)
1278          g2.setStroke(bs3);
1279        else
1280          g2.setStroke(bs1);
1281
1282        for (int i = 0; i < mP08Points.length - 1; i++) {
1283          lPP08.setLine(mP08Points[i], mP08Points[i+1]);
1284          g2.draw(lPP08);
1285        }
1286
1287        // Produkt 09
1288        mP09Points = new Point2D[14];
1289        mP09Points[0]  = new Point2D.Double(300, 615);
1290        mP09Points[1]  = new Point2D.Double(350, 599);
1291        mP09Points[2]  = new Point2D.Double(400, 700);
1292        mP09Points[3]  = new Point2D.Double(450, 645);
1293        mP09Points[4]  = new Point2D.Double(500, 551);
1294        mP09Points[5]  = new Point2D.Double(550, 739);
1295        mP09Points[6]  = new Point2D.Double(600, 707);
1296        mP09Points[7]  = new Point2D.Double(650, 719);
1297        mP09Points[8]  = new Point2D.Double(700, 725);
1298        mP09Points[9]  = new Point2D.Double(750, 720);
1299        mP09Points[10] = new Point2D.Double(800, 720);
1300        mP09Points[11] = new Point2D.Double(850, 695);
1301        mP09Points[12] = new Point2D.Double(900, 565);
1302        mP09Points[13] = new Point2D.Double(950, 645);
```





```
1303
1304        g2.setPaint(c09);
1305
1306        if (produkt == 9)
1307          g2.setStroke(bs3);
1308        else
1309          g2.setStroke(bs1);
1310
1311        for (int i = 0; i < mP09Points.length - 1; i++) {
1312          lPP09.setLine(mP09Points[i], mP09Points[i+1]);
1313          g2.draw(lPP09);
1314        }
1315
1316        // Produkt 10
1317        mP10Points = new Point2D[14];
1318        mP10Points[0]  = new Point2D.Double(300, 585);
1319        mP10Points[1]  = new Point2D.Double(350, 559);
1320        mP10Points[2]  = new Point2D.Double(400, 727);
1321        mP10Points[3]  = new Point2D.Double(450, 645);
1322        mP10Points[4]  = new Point2D.Double(500, 635);
1323        mP10Points[5]  = new Point2D.Double(550, 711);
1324        mP10Points[6]  = new Point2D.Double(600, 699);
1325        mP10Points[7]  = new Point2D.Double(650, 691);
1326        mP10Points[8]  = new Point2D.Double(700, 705);
1327        mP10Points[9]  = new Point2D.Double(750, 710);
1328        mP10Points[10] = new Point2D.Double(800, 720);
1329        mP10Points[11] = new Point2D.Double(850, 695);
1330        mP10Points[12] = new Point2D.Double(900, 725);
1331        mP10Points[13] = new Point2D.Double(950, 645);
1332
1333        g2.setPaint(c10);
1334
1335        if (produkt == 10)
1336          g2.setStroke(bs3);
1337        else
1338          g2.setStroke(bs1);
1339
1340        for (int i = 0; i < mP10Points.length - 1; i++) {
1341          lPP10.setLine(mP10Points[i], mP10Points[i+1]);
1342          g2.draw(lPP10);
1343        }
1344
1345        // Produkt 11
1346        mP11Points = new Point2D[14];
1347        mP11Points[0]  = new Point2D.Double(300, 635);
1348        mP11Points[1]  = new Point2D.Double(350, 679);
1349        mP11Points[2]  = new Point2D.Double(400, 744);
1350        mP11Points[3]  = new Point2D.Double(450, 645);
1351        mP11Points[4]  = new Point2D.Double(500, 635);
1352        mP11Points[5]  = new Point2D.Double(550, 739);
1353        mP11Points[6]  = new Point2D.Double(600, 727);
1354        mP11Points[7]  = new Point2D.Double(650, 719);
1355        mP11Points[8]  = new Point2D.Double(700, 745);
1356        mP11Points[9]  = new Point2D.Double(750, 740);
1357        mP11Points[10] = new Point2D.Double(800, 720);
1358        mP11Points[11] = new Point2D.Double(850, 695);
1359        mP11Points[12] = new Point2D.Double(900, 685);
1360        mP11Points[13] = new Point2D.Double(950, 645);
1361
1362        g2.setPaint(c11);
1363
1364        if (produkt == 11)
```

```
1365          g2.setStroke(bs3);
1366        else
1367          g2.setStroke(bs1);
1368
1369        for (int i = 0; i < mP11Points.length - 1; i++) {
1370          lPP11.setLine(mP11Points[i], mP11Points[i+1]);
1371          g2.draw(lPP11);
1372        }
1373
1374        // Produkt 12
1375        mP12Points = new Point2D[14];
1376        mP12Points[0]  = new Point2D.Double(300, 655);
1377        mP12Points[1]  = new Point2D.Double(350, 695);
1378        mP12Points[2]  = new Point2D.Double(400, 718);
1379        mP12Points[3]  = new Point2D.Double(450, 645);
1380        mP12Points[4]  = new Point2D.Double(500, 551);
1381        mP12Points[5]  = new Point2D.Double(550, 739);
1382        mP12Points[6]  = new Point2D.Double(600, 699);
1383        mP12Points[7]  = new Point2D.Double(650, 675);
1384        mP12Points[8]  = new Point2D.Double(700, 665);
1385        mP12Points[9]  = new Point2D.Double(750, 720);
1386        mP12Points[10] = new Point2D.Double(800, 620);
1387        mP12Points[11] = new Point2D.Double(850, 695);
1388        mP12Points[12] = new Point2D.Double(900, 645);
1389        mP12Points[13] = new Point2D.Double(950, 645);
1390
1391        g2.setPaint(c12);
1392
1393        if (produkt == 12)
1394          g2.setStroke(bs3);
1395        else
1396          g2.setStroke(bs1);
1397
1398        for (int i = 0; i < mP12Points.length - 1; i++) {
1399          lPP12.setLine(mP12Points[i], mP12Points[i+1]);
1400          g2.draw(lPP12);
1401        }
1402
1403        // Produkt 13
1404        mP13Points = new Point2D[14];
1405        mP13Points[0]  = new Point2D.Double(300, 625);
1406        mP13Points[1]  = new Point2D.Double(350, 655);
1407        mP13Points[2]  = new Point2D.Double(400, 546);
1408        mP13Points[3]  = new Point2D.Double(450, 645);
1409        mP13Points[4]  = new Point2D.Double(500, 551);
1410        mP13Points[5]  = new Point2D.Double(550, 739);
1411        mP13Points[6]  = new Point2D.Double(600, 663);
1412        mP13Points[7]  = new Point2D.Double(650, 711);
1413        mP13Points[8]  = new Point2D.Double(700, 545);
1414        mP13Points[9]  = new Point2D.Double(750, 690);
1415        mP13Points[10] = new Point2D.Double(800, 570);
1416        mP13Points[11] = new Point2D.Double(850, 695);
1417        mP13Points[12] = new Point2D.Double(900, 605);
1418        mP13Points[13] = new Point2D.Double(950, 645);
1419
1420        g2.setPaint(c13);
1421
1422        if (produkt == 13)
1423          g2.setStroke(bs3);
1424        else
1425          g2.setStroke(bs1);
1426
```





```
1427        for (int i = 0; i < mP13Points.length - 1; i++) {
1428          lPP13.setLine(mP13Points[i], mP13Points[i+1]);
1429          g2.draw(lPP13);
1430        }
1431
1432        // Produkt 14
1433        mP14Points = new Point2D[14];
1434        mP14Points[0]  = new Point2D.Double(300, 705);
1435        mP14Points[1]  = new Point2D.Double(350, 629);
1436        mP14Points[2]  = new Point2D.Double(400, 696);
1437        mP14Points[3]  = new Point2D.Double(450, 645);
1438        mP14Points[4]  = new Point2D.Double(500, 551);
1439        mP14Points[5]  = new Point2D.Double(550, 731);
1440        mP14Points[6]  = new Point2D.Double(600, 703);
1441        mP14Points[7]  = new Point2D.Double(650, 715);
1442        mP14Points[8]  = new Point2D.Double(700, 585);
1443        mP14Points[9]  = new Point2D.Double(750, 720);
1444        mP14Points[10] = new Point2D.Double(800, 670);
1445        mP14Points[11] = new Point2D.Double(850, 695);
1446        mP14Points[12] = new Point2D.Double(900, 725);
1447        mP14Points[13] = new Point2D.Double(950, 645);
1448
1449        g2.setPaint(c14);
1450
1451        if (produkt == 14)
1452          g2.setStroke(bs3);
1453        else
1454          g2.setStroke(bs1);
1455
1456        for (int i = 0; i < mP14Points.length - 1; i++) {
1457          lPP14.setLine(mP14Points[i], mP14Points[i+1]);
1458          g2.draw(lPP14);
1459        }
1460
1461        // Produkt 15
1462        mP15Points = new Point2D[14];
1463        mP15Points[0]  = new Point2D.Double(300, 675);
1464        mP15Points[1]  = new Point2D.Double(350, 687);
1465        mP15Points[2]  = new Point2D.Double(400, 700);
1466        mP15Points[3]  = new Point2D.Double(450, 645);
1467        mP15Points[4]  = new Point2D.Double(500, 739);
1468        mP15Points[5]  = new Point2D.Double(550, 735);
1469        mP15Points[6]  = new Point2D.Double(600, 727);
1470        mP15Points[7]  = new Point2D.Double(650, 711);
1471        mP15Points[8]  = new Point2D.Double(700, 725);
1472        mP15Points[9]  = new Point2D.Double(750, 740);
1473        mP15Points[10] = new Point2D.Double(800, 720);
1474        mP15Points[11] = new Point2D.Double(850, 695);
1475        mP15Points[12] = new Point2D.Double(900, 645);
1476        mP15Points[13] = new Point2D.Double(950, 645);
1477
1478        g2.setPaint(c15);
1479
1480        if (produkt == 15)
1481          g2.setStroke(bs3);
1482        else
1483          g2.setStroke(bs1);
1484
1485        for (int i = 0; i < mP15Points.length - 1; i++) {
1486          lPP15.setLine(mP15Points[i], mP15Points[i+1]);
1487          g2.draw(lPP15);
1488        }
```

```
1489
1490        // Produkt 16
1491        mP16Points = new Point2D[14];
1492        mP16Points[0]  = new Point2D.Double(300, 685);
1493        mP16Points[1]  = new Point2D.Double(350, 680);
1494        mP16Points[2]  = new Point2D.Double(400, 700);
1495        mP16Points[3]  = new Point2D.Double(450, 645);
1496        mP16Points[4]  = new Point2D.Double(500, 551);
1497        mP16Points[5]  = new Point2D.Double(550, 739);
1498        mP16Points[6]  = new Point2D.Double(600, 715);
1499        mP16Points[7]  = new Point2D.Double(650, 695);
1500        mP16Points[8]  = new Point2D.Double(700, 705);
1501        mP16Points[9]  = new Point2D.Double(750, 730);
1502        mP16Points[10] = new Point2D.Double(800, 720);
1503        mP16Points[11] = new Point2D.Double(850, 595);
1504        mP16Points[12] = new Point2D.Double(900, 685);
1505        mP16Points[13] = new Point2D.Double(950, 645);
1506
1507        g2.setPaint(c16);
1508
1509        if (produkt == 16)
1510          g2.setStroke(bs3);
1511        else
1512          g2.setStroke(bs1);
1513
1514        for (int i = 0; i < mP16Points.length - 1; i++) {
1515          lPP16.setLine(mP16Points[i], mP16Points[i+1]);
1516          g2.draw(lPP16);
1517        }
1518
1519      } // end if produktL == 3
1520
1521      if (knappnr == 2){
1522
1523        // Parallellkoordinater (produktverdier)
1524        // Produkt 02
1525        mP02_kn2Points = new Point2D[15];
1526        mP02_kn2Points[0]  = new Point2D.Double(300, 685);
1527        mP02_kn2Points[1]  = new Point2D.Double(350, 630);
1528        mP02_kn2Points[2]  = new Point2D.Double(400, 702);
1529        mP02_kn2Points[3]  = new Point2D.Double(450, 645);
1530        mP02_kn2Points[4]  = new Point2D.Double(475, 682); // ekstra punkt
1531        mP02_kn2Points[5]  = new Point2D.Double(500, 739);
1532        mP02_kn2Points[6]  = new Point2D.Double(550, 739);
1533        mP02_kn2Points[7]  = new Point2D.Double(600, 715);
1534        mP02_kn2Points[8]  = new Point2D.Double(650, 719);
1535        mP02_kn2Points[9]  = new Point2D.Double(700, 665);
1536        mP02_kn2Points[10] = new Point2D.Double(750, 740);
1537        mP02_kn2Points[11] = new Point2D.Double(800, 720);
1538        mP02_kn2Points[12] = new Point2D.Double(850, 695);
1539        mP02_kn2Points[13] = new Point2D.Double(900, 645);
1540        mP02_kn2Points[14] = new Point2D.Double(950, 645);
1541
1542        g2.setPaint(c02);
1543
1544        if (produkt == 2)
1545          g2.setStroke(bs3);
1546        else
1547          g2.setStroke(bs1);
1548
1549        for (int i = 0; i < mP02_kn2Points.length - 1; i++) {
1550          lPP02_kn2.setLine(mP02_kn2Points[i], mP02_kn2Points[i+1]);
```





```
1551          g2.draw(lPP02_kn2);
1552        }
1553
1554        // Produkt 03
1555        mP03_kn2Points = new Point2D[14];
1556        mP03_kn2Points[0]  = new Point2D.Double(300, 675);
1557        mP03_kn2Points[1]  = new Point2D.Double(350, 673);
1558        mP03_kn2Points[2]  = new Point2D.Double(400, 726);
1559        mP03_kn2Points[3]  = new Point2D.Double(450, 645);
1560        mP03_kn2Points[4]  = new Point2D.Double(500, 651);
1561        mP03_kn2Points[5]  = new Point2D.Double(550, 727);
1562        mP03_kn2Points[6]  = new Point2D.Double(600, 643);
1563        mP03_kn2Points[7]  = new Point2D.Double(650, 571);
1564        mP03_kn2Points[8]  = new Point2D.Double(700, 745);
1565        mP03_kn2Points[9]  = new Point2D.Double(750, 630);
1566        mP03_kn2Points[10] = new Point2D.Double(800, 720);
1567        mP03_kn2Points[11] = new Point2D.Double(850, 695);
1568        mP03_kn2Points[12] = new Point2D.Double(900, 605);
1569        mP03_kn2Points[13] = new Point2D.Double(950, 645);
1570
1571        g2.setPaint(c03);
1572
1573        if (produkt == 3)
1574          g2.setStroke(bs3);
1575        else
1576          g2.setStroke(bs1);
1577
1578        for (int i = 0; i < mP03_kn2Points.length - 1; i++) {
1579          lPP03_kn2.setLine(mP03_kn2Points[i], mP03_kn2Points[i+1]);
1580          g2.draw(lPP03_kn2);
1581        }
1582
1583        // Produkt 04
1584        mP04_kn2Points = new Point2D[14];
1585        mP04_kn2Points[0]  = new Point2D.Double(300, 705);
1586        mP04_kn2Points[1]  = new Point2D.Double(350, 688);
1587        mP04_kn2Points[2]  = new Point2D.Double(400, 704);
1588        mP04_kn2Points[3]  = new Point2D.Double(450, 645);
1589        mP04_kn2Points[4]  = new Point2D.Double(500, 739);
1590        mP04_kn2Points[5]  = new Point2D.Double(550, 551);
1591        mP04_kn2Points[6]  = new Point2D.Double(600, 563);
1592        mP04_kn2Points[7]  = new Point2D.Double(650, 719);
1593        mP04_kn2Points[8]  = new Point2D.Double(700, 605);
1594        mP04_kn2Points[9]  = new Point2D.Double(750, 550);
1595        mP04_kn2Points[10] = new Point2D.Double(800, 720);
1596        mP04_kn2Points[11] = new Point2D.Double(850, 695);
1597        mP04_kn2Points[12] = new Point2D.Double(900, 645);
1598        mP04_kn2Points[13] = new Point2D.Double(950, 645);
1599
1600        g2.setPaint(c04);
1601
1602        if (produkt == 4)
1603          g2.setStroke(bs3);
1604        else
1605          g2.setStroke(bs1);
1606
1607        for (int i = 0; i < mP04_kn2Points.length - 1; i++) {
1608          lPP04_kn2.setLine(mP04_kn2Points[i], mP04_kn2Points[i+1]);
1609          g2.draw(lPP04_kn2);
1610        }
1611
1612        // Produkt 05
```

```
1613        mP05_kn2Points = new Point2D[15];
1614        mP05_kn2Points[0]  = new Point2D.Double(300, 665);
1615        mP05_kn2Points[1]  = new Point2D.Double(350, 646);
1616        mP05_kn2Points[2]  = new Point2D.Double(400, 673);
1617        mP05_kn2Points[3]  = new Point2D.Double(450, 645);
1618        mP05_kn2Points[4]  = new Point2D.Double(470, 575); // ekstra punkt
1619        mP05_kn2Points[5]  = new Point2D.Double(500, 551);
1620        mP05_kn2Points[6]  = new Point2D.Double(550, 739);
1621        mP05_kn2Points[7]  = new Point2D.Double(600, 719);
1622        mP05_kn2Points[8]  = new Point2D.Double(650, 719);
1623        mP05_kn2Points[9]  = new Point2D.Double(700, 585);
1624        mP05_kn2Points[10] = new Point2D.Double(750, 730);
1625        mP05_kn2Points[11] = new Point2D.Double(800, 720);
1626        mP05_kn2Points[12] = new Point2D.Double(850, 695);
1627        mP05_kn2Points[13] = new Point2D.Double(900, 645);
1628        mP05_kn2Points[14] = new Point2D.Double(950, 645);
1629
1630        g2.setPaint(c05);
1631
1632        if (produkt == 5)
1633          g2.setStroke(bs3);
1634        else
1635          g2.setStroke(bs1);
1636
1637        for (int i = 0; i < mP05_kn2Points.length - 1; i++) {
1638          lPP05_kn2.setLine(mP05_kn2Points[i], mP05_kn2Points[i+1]);
1639          g2.draw(lPP05_kn2);
1640        }
1641
1642        // Produkt 06
1643        mP06_kn2Points = new Point2D[15];
1644        mP06_kn2Points[0]  = new Point2D.Double(300, 665);
1645        mP06_kn2Points[1]  = new Point2D.Double(350, 644);
1646        mP06_kn2Points[2]  = new Point2D.Double(400, 703);
1647        mP06_kn2Points[3]  = new Point2D.Double(450, 645);
1648        mP06_kn2Points[4]  = new Point2D.Double(475, 575); // ekstra punkt
1649        mP06_kn2Points[5]  = new Point2D.Double(500, 551);
1650        mP06_kn2Points[6]  = new Point2D.Double(550, 739);
1651        mP06_kn2Points[7]  = new Point2D.Double(600, 603);
1652        mP06_kn2Points[8]  = new Point2D.Double(650, 675);
1653        mP06_kn2Points[9]  = new Point2D.Double(700, 705);
1654        mP06_kn2Points[10] = new Point2D.Double(750, 550);
1655        mP06_kn2Points[11] = new Point2D.Double(800, 720);
1656        mP06_kn2Points[12] = new Point2D.Double(850, 695);
1657        mP06_kn2Points[13] = new Point2D.Double(900, 685);
1658        mP06_kn2Points[14] = new Point2D.Double(950, 645);
1659
1660        g2.setPaint(c06);
1661
1662        if (produkt == 6)
1663          g2.setStroke(bs3);
1664        else
1665          g2.setStroke(bs1);
1666
1667        for (int i = 0; i < mP06_kn2Points.length - 1; i++) {
1668          lPP06_kn2.setLine(mP06_kn2Points[i], mP06_kn2Points[i+1]);
1669          g2.draw(lPP06_kn2);
1670        }
1671
1672        // Produkt 07
1673        mP07_kn2Points = new Point2D[15];
1674        mP07_kn2Points[0]  = new Point2D.Double(300, 645);
```





```
1675        mP07_kn2Points[1]  = new Point2D.Double(350, 671);
1676        mP07_kn2Points[2]  = new Point2D.Double(400, 715);
1677        mP07_kn2Points[3]  = new Point2D.Double(450, 645);
1678        mP07_kn2Points[4]  = new Point2D.Double(475, 635); // ekstra punkt
1679        mP07_kn2Points[5]  = new Point2D.Double(500, 635);
1680        mP07_kn2Points[6]  = new Point2D.Double(550, 731);
1681        mP07_kn2Points[7]  = new Point2D.Double(600, 675);
1682        mP07_kn2Points[8]  = new Point2D.Double(650, 695);
1683        mP07_kn2Points[9]  = new Point2D.Double(700, 745);
1684        mP07_kn2Points[10] = new Point2D.Double(750, 670);
1685        mP07_kn2Points[11] = new Point2D.Double(800, 720);
1686        mP07_kn2Points[12] = new Point2D.Double(850, 695);
1687        mP07_kn2Points[13] = new Point2D.Double(900, 645);
1688        mP07_kn2Points[14] = new Point2D.Double(950, 645);
1689
1690        g2.setPaint(c07);
1691
1692        if (produkt == 7)
1693          g2.setStroke(bs3);
1694        else
1695          g2.setStroke(bs1);
1696
1697        for (int i = 0; i < mP07_kn2Points.length - 1; i++) {
1698          lPP07_kn2.setLine(mP07_kn2Points[i], mP07_kn2Points[i+1]);
1699          g2.draw(lPP07_kn2);
1700        }
1701
1702        // Produkt 08
1703        mP08_kn2Points = new Point2D[15];
1704        mP08_kn2Points[0]  = new Point2D.Double(300, 675);
1705        mP08_kn2Points[1]  = new Point2D.Double(350, 731);
1706        mP08_kn2Points[2]  = new Point2D.Double(400, 712);
1707        mP08_kn2Points[3]  = new Point2D.Double(450, 645);
1708        mP08_kn2Points[4]  = new Point2D.Double(475, 585); // ekstra punkt
1709        mP08_kn2Points[5]  = new Point2D.Double(500, 551);
1710        mP08_kn2Points[6]  = new Point2D.Double(550, 739);
1711        mP08_kn2Points[7]  = new Point2D.Double(600, 691);
1712        mP08_kn2Points[8]  = new Point2D.Double(650, 715);
1713        mP08_kn2Points[9]  = new Point2D.Double(700, 665);
1714        mP08_kn2Points[10] = new Point2D.Double(750, 700);
1715        mP08_kn2Points[11] = new Point2D.Double(800, 720);
1716        mP08_kn2Points[12] = new Point2D.Double(850, 695);
1717        mP08_kn2Points[13] = new Point2D.Double(900, 605);
1718        mP08_kn2Points[14] = new Point2D.Double(950, 645);
1719
1720        g2.setPaint(c08);
1721
1722        if (produkt == 8)
1723          g2.setStroke(bs3);
1724        else
1725          g2.setStroke(bs1);
1726
1727        for (int i = 0; i < mP08_kn2Points.length - 1; i++) {
1728          lPP08_kn2.setLine(mP08_kn2Points[i], mP08_kn2Points[i+1]);
1729          g2.draw(lPP08_kn2);
1730        }
1731
1732        // Produkt 09
1733        mP09_kn2Points = new Point2D[15];
1734        mP09_kn2Points[0]  = new Point2D.Double(300, 615);
1735        mP09_kn2Points[1]  = new Point2D.Double(350, 599);
1736        mP09_kn2Points[2]  = new Point2D.Double(400, 700);
```

```
1737        mP09_kn2Points[3]  = new Point2D.Double(450, 645);
1738        mP09_kn2Points[4]  = new Point2D.Double(475, 595); // ekstra punkt
1739        mP09_kn2Points[5]  = new Point2D.Double(500, 551);
1740        mP09_kn2Points[6]  = new Point2D.Double(550, 739);
1741        mP09_kn2Points[7]  = new Point2D.Double(600, 707);
1742        mP09_kn2Points[8]  = new Point2D.Double(650, 719);
1743        mP09_kn2Points[9]  = new Point2D.Double(700, 725);
1744        mP09_kn2Points[10] = new Point2D.Double(750, 720);
1745        mP09_kn2Points[11] = new Point2D.Double(800, 720);
1746        mP09_kn2Points[12] = new Point2D.Double(850, 695);
1747        mP09_kn2Points[13] = new Point2D.Double(900, 565);
1748        mP09_kn2Points[14] = new Point2D.Double(950, 645);
1749
1750        g2.setPaint(c09);
1751
1752        if (produkt == 9)
1753          g2.setStroke(bs3);
1754        else
1755          g2.setStroke(bs1);
1756
1757        for (int i = 0; i < mP09_kn2Points.length - 1; i++) {
1758          lPP09_kn2.setLine(mP09_kn2Points[i], mP09_kn2Points[i+1]);
1759          g2.draw(lPP09_kn2);
1760        }
1761
1762        // Produkt 10
1763        mP10_kn2Points = new Point2D[14];
1764        mP10_kn2Points[0]  = new Point2D.Double(300, 585);
1765        mP10_kn2Points[1]  = new Point2D.Double(350, 559);
1766        mP10_kn2Points[2]  = new Point2D.Double(400, 727);
1767        mP10_kn2Points[3]  = new Point2D.Double(450, 645);
1768        mP10_kn2Points[4]  = new Point2D.Double(500, 635);
1769        mP10_kn2Points[5]  = new Point2D.Double(550, 711);
1770        mP10_kn2Points[6]  = new Point2D.Double(600, 699);
1771        mP10_kn2Points[7]  = new Point2D.Double(650, 691);
1772        mP10_kn2Points[8]  = new Point2D.Double(700, 705);
1773        mP10_kn2Points[9]  = new Point2D.Double(750, 710);
1774        mP10_kn2Points[10] = new Point2D.Double(800, 720);
1775        mP10_kn2Points[11] = new Point2D.Double(850, 695);
1776        mP10_kn2Points[12] = new Point2D.Double(900, 725);
1777        mP10_kn2Points[13] = new Point2D.Double(950, 645);
1778
1779        g2.setPaint(c10);
1780
1781        if (produkt == 10)
1782          g2.setStroke(bs3);
1783        else
1784          g2.setStroke(bs1);
1785
1786        for (int i = 0; i < mP10_kn2Points.length - 1; i++) {
1787          lPP10_kn2.setLine(mP10_kn2Points[i], mP10_kn2Points[i+1]);
1788          g2.draw(lPP10_kn2);
1789        }
1790
1791        // Produkt 11
1792        mP11_kn2Points = new Point2D[15];
1793        mP11_kn2Points[0]  = new Point2D.Double(300, 635);
1794        mP11_kn2Points[1]  = new Point2D.Double(350, 679);
1795        mP11_kn2Points[2]  = new Point2D.Double(400, 744);
1796        mP11_kn2Points[3]  = new Point2D.Double(450, 645);
1797        mP11_kn2Points[4]  = new Point2D.Double(475, 645); // ekstra punkt
1798        mP11_kn2Points[5]  = new Point2D.Double(500, 635);
```





```
1799        mP11_kn2Points[6]  = new Point2D.Double(550, 739);
1800        mP11_kn2Points[7]  = new Point2D.Double(600, 727);
1801        mP11_kn2Points[8]  = new Point2D.Double(650, 719);
1802        mP11_kn2Points[9]  = new Point2D.Double(700, 745);
1803        mP11_kn2Points[10] = new Point2D.Double(750, 740);
1804        mP11_kn2Points[11] = new Point2D.Double(800, 720);
1805        mP11_kn2Points[12] = new Point2D.Double(850, 695);
1806        mP11_kn2Points[13] = new Point2D.Double(900, 685);
1807        mP11_kn2Points[14] = new Point2D.Double(950, 645);
1808
1809        g2.setPaint(c11);
1810
1811        if (produkt == 11)
1812          g2.setStroke(bs3);
1813        else
1814          g2.setStroke(bs1);
1815
1816        for (int i = 0; i < mP11_kn2Points.length - 1; i++) {
1817          lPP11_kn2.setLine(mP11_kn2Points[i], mP11_kn2Points[i+1]);
1818          g2.draw(lPP11_kn2);
1819        }
1820
1821        // Produkt 12
1822        mP12_kn2Points = new Point2D[15];
1823        mP12_kn2Points[0]  = new Point2D.Double(300, 655);
1824        mP12_kn2Points[1]  = new Point2D.Double(350, 695);
1825        mP12_kn2Points[2]  = new Point2D.Double(400, 718);
1826        mP12_kn2Points[3]  = new Point2D.Double(450, 645);
1827        mP12_kn2Points[4]  = new Point2D.Double(475, 605); // ekstra punkt
1828        mP12_kn2Points[5]  = new Point2D.Double(500, 551);
1829        mP12_kn2Points[6]  = new Point2D.Double(550, 739);
1830        mP12_kn2Points[7]  = new Point2D.Double(600, 699);
1831        mP12_kn2Points[8]  = new Point2D.Double(650, 675);
1832        mP12_kn2Points[9]  = new Point2D.Double(700, 665);
1833        mP12_kn2Points[10] = new Point2D.Double(750, 720);
1834        mP12_kn2Points[11] = new Point2D.Double(800, 620);
1835        mP12_kn2Points[12] = new Point2D.Double(850, 695);
1836        mP12_kn2Points[13] = new Point2D.Double(900, 645);
1837        mP12_kn2Points[14] = new Point2D.Double(950, 645);
1838
1839        g2.setPaint(c12);
1840
1841        if (produkt == 12)
1842          g2.setStroke(bs3);
1843        else
1844          g2.setStroke(bs1);
1845
1846        for (int i = 0; i < mP12_kn2Points.length - 1; i++) {
1847          lPP12_kn2.setLine(mP12_kn2Points[i], mP12_kn2Points[i+1]);
1848          g2.draw(lPP12_kn2);
1849        }
1850
1851        // Produkt 13
1852        mP13_kn2Points = new Point2D[15];
1853        mP13_kn2Points[0]  = new Point2D.Double(300, 625);
1854        mP13_kn2Points[1]  = new Point2D.Double(350, 655);
1855        mP13_kn2Points[2]  = new Point2D.Double(400, 546);
1856        mP13_kn2Points[3]  = new Point2D.Double(450, 645);
1857        mP13_kn2Points[4]  = new Point2D.Double(475, 615); // ekstra punkt
1858        mP13_kn2Points[5]  = new Point2D.Double(500, 551);
1859        mP13_kn2Points[6]  = new Point2D.Double(550, 739);
1860        mP13_kn2Points[7]  = new Point2D.Double(600, 663);
```

```
1861        mP13_kn2Points[8]  = new Point2D.Double(650, 711);
1862        mP13_kn2Points[9]  = new Point2D.Double(700, 545);
1863        mP13_kn2Points[10] = new Point2D.Double(750, 690);
1864        mP13_kn2Points[11] = new Point2D.Double(800, 570);
1865        mP13_kn2Points[12] = new Point2D.Double(850, 695);
1866        mP13_kn2Points[13] = new Point2D.Double(900, 605);
1867        mP13_kn2Points[14] = new Point2D.Double(950, 645);
1868
1869        g2.setPaint(c13);
1870
1871        if (produkt == 13)
1872          g2.setStroke(bs3);
1873        else
1874          g2.setStroke(bs1);
1875
1876        for (int i = 0; i < mP13_kn2Points.length - 1; i++) {
1877          lPP13_kn2.setLine(mP13_kn2Points[i], mP13_kn2Points[i+1]);
1878          g2.draw(lPP13_kn2);
1879        }
1880
1881        // Produkt 14
1882        mP14_kn2Points = new Point2D[15];
1883        mP14_kn2Points[0]  = new Point2D.Double(300, 705);
1884        mP14_kn2Points[1]  = new Point2D.Double(350, 629);
1885        mP14_kn2Points[2]  = new Point2D.Double(400, 696);
1886        mP14_kn2Points[3]  = new Point2D.Double(450, 645);
1887        mP14_kn2Points[4]  = new Point2D.Double(475, 625); // ekstra punkt
1888        mP14_kn2Points[5]  = new Point2D.Double(500, 551);
1889        mP14_kn2Points[6]  = new Point2D.Double(550, 731);
1890        mP14_kn2Points[7]  = new Point2D.Double(600, 703);
1891        mP14_kn2Points[8]  = new Point2D.Double(650, 715);
1892        mP14_kn2Points[9]  = new Point2D.Double(700, 585);
1893        mP14_kn2Points[10] = new Point2D.Double(750, 720);
1894        mP14_kn2Points[11] = new Point2D.Double(800, 670);
1895        mP14_kn2Points[12] = new Point2D.Double(850, 695);
1896        mP14_kn2Points[13] = new Point2D.Double(900, 725);
1897        mP14_kn2Points[14] = new Point2D.Double(950, 645);
1898
1899        g2.setPaint(c14);
1900
1901        if (produkt == 14)
1902          g2.setStroke(bs3);
1903        else
1904          g2.setStroke(bs1);
1905
1906        for (int i = 0; i < mP14_kn2Points.length - 1; i++) {
1907          lPP14_kn2.setLine(mP14_kn2Points[i], mP14_kn2Points[i+1]);
1908          g2.draw(lPP14_kn2);
1909        }
1910
1911        // Produkt 15
1912        mP15_kn2Points = new Point2D[15];
1913        mP15_kn2Points[0]  = new Point2D.Double(300, 675);
1914        mP15_kn2Points[1]  = new Point2D.Double(350, 687);
1915        mP15_kn2Points[2]  = new Point2D.Double(400, 700);
1916        mP15_kn2Points[3]  = new Point2D.Double(450, 645);
1917        mP15_kn2Points[4]  = new Point2D.Double(475, 702); // ekstra punkt
1918        mP15_kn2Points[5]  = new Point2D.Double(500, 739);
1919        mP15_kn2Points[6]  = new Point2D.Double(550, 735);
1920        mP15_kn2Points[7]  = new Point2D.Double(600, 727);
1921        mP15_kn2Points[8]  = new Point2D.Double(650, 711);
1922        mP15_kn2Points[9]  = new Point2D.Double(700, 725);
```





```
1923        mP15_kn2Points[10] = new Point2D.Double(750, 740);
1924        mP15_kn2Points[11] = new Point2D.Double(800, 720);
1925        mP15_kn2Points[12] = new Point2D.Double(850, 695);
1926        mP15_kn2Points[13] = new Point2D.Double(900, 645);
1927        mP15_kn2Points[14] = new Point2D.Double(950, 645);
1928
1929        g2.setPaint(c15);
1930
1931        if (produkt == 15)
1932          g2.setStroke(bs3);
1933        else
1934          g2.setStroke(bs1);
1935
1936        for (int i = 0; i < mP15_kn2Points.length - 1; i++) {
1937          lPP15_kn2.setLine(mP15_kn2Points[i], mP15_kn2Points[i+1]);
1938          g2.draw(lPP15_kn2);
1939        }
1940
1941        // Produkt 16
1942        mP16_kn2Points = new Point2D[15];
1943        mP16_kn2Points[0]  = new Point2D.Double(300, 685);
1944        mP16_kn2Points[1]  = new Point2D.Double(350, 680);
1945        mP16_kn2Points[2]  = new Point2D.Double(400, 700);
1946        mP16_kn2Points[3]  = new Point2D.Double(450, 645);
1947        mP16_kn2Points[4]  = new Point2D.Double(480, 625); // ekstra punkt
1948        mP16_kn2Points[5]  = new Point2D.Double(500, 551);
1949        mP16_kn2Points[6]  = new Point2D.Double(550, 739);
1950        mP16_kn2Points[7]  = new Point2D.Double(600, 715);
1951        mP16_kn2Points[8]  = new Point2D.Double(650, 695);
1952        mP16_kn2Points[9]  = new Point2D.Double(700, 705);
1953        mP16_kn2Points[10] = new Point2D.Double(750, 730);
1954        mP16_kn2Points[11] = new Point2D.Double(800, 720);
1955        mP16_kn2Points[12] = new Point2D.Double(850, 595);
1956        mP16_kn2Points[13] = new Point2D.Double(900, 685);
1957        mP16_kn2Points[14] = new Point2D.Double(950, 645);
1958
1959        g2.setPaint(c16);
1960
1961        if (produkt == 16)
1962          g2.setStroke(bs3);
1963        else
1964          g2.setStroke(bs1);
1965
1966        for (int i = 0; i < mP16_kn2Points.length - 1; i++) {
1967          lPP16_kn2.setLine(mP16_kn2Points[i], mP16_kn2Points[i+1]);
1968          g2.draw(lPP16_kn2);
1969        }
1970
1971      } // end if knappnr == 2
1972
1973      if (produktL == 3 || prosess == 5 || knappnr > 0) {
1974
1975      // Produktnivå, sentrert
1976        if (produkt == 1) {
1977          g2.setPaint(c01);
1978        }
1979
1980        if (produkt == 2) {
1981          g2.setPaint(c02);
1982        }
1983
1984        if (produkt == 3) {
```

```
1985          g2.setPaint(c03);
1986        }
1987
1988        if (produkt == 4) {
1989          g2.setPaint(c04);
1990        }
1991
1992        if (produkt == 5) {
1993          g2.setPaint(c05);
1994        }
1995
1996        if (produkt == 6) {
1997          g2.setPaint(c06);
1998        }
1999
2000        if (produkt == 7) {
2001          g2.setPaint(c07);
2002        }
2003
2004        if (produkt == 8) {
2005          g2.setPaint(c08);
2006        }
2007
2008        if (produkt == 9) {
2009          g2.setPaint(c09);
2010        }
2011
2012        if (produkt == 10) {
2013          g2.setPaint(c10);
2014        }
2015
2016        if (produkt == 11) {
2017          g2.setPaint(c11);
2018        }
2019
2020        if (produkt == 12) {
2021          g2.setPaint(c12);
2022        }
2023
2024        if (produkt == 13) {
2025          g2.setPaint(c13);
2026        }
2027
2028        if (produkt == 14) {
2029          g2.setPaint(c14);
2030        }
2031
2032        if (produkt == 15) {
2033          g2.setPaint(c15);
2034        }
2035
2036        if (produkt == 16) {
2037          g2.setPaint(c16);
2038        }
2039
2040      }
2041
2042      // Prosess 5
2043      if (prosess == 5) {
2044        g2.setPaint(c1);
2045        g2.setStroke(bs1);
2046      }
```





```
2047      else
2048        g2.setStroke(bs3);
2049
2050      // Tegner midtstilt sirkel (produktplan)
2051      Etmp = new Ellipse2D.Double(E1,E2,El,El);
2052      g2.draw(Etmp);
2053
2054      // Nullstiller sirkel
2055      E1 = 0; E2 = 0; El = 0;
2056
2057      // Tegner produkt-view
2058      g2.setStroke(bs3);
2059
2060      g2.setPaint(cProdukt);
2061      produktBkg = new RoundRectangle2D.Double(290, 80, 210, 375, 10, 10);
2062      g2.draw(produktBkg);
2063
2064      // A2D6
2065      if (produktL == 6)
2066        g2.setPaint(c1);
2067      else
2068        g2.setPaint(cStd);
2069
2070      A2D1_1 = new Arc2D.Double(300,125,150,25,0,-180,Arc2D.OPEN);
2071      g2.draw(A2D1_1);
2072      A2D1_2 = new Arc2D.Double(300,100,150,25,0,360,Arc2D.OPEN);
2073      g2.draw(A2D1_2);
2074      Line2D lE2D1_1 = new Line2D.Double(300, 112.5, 300, 137.5);
2075      g2.draw(lE2D1_1);
2076      Line2D lE2D1_2 = new Line2D.Double(450, 112.5, 450, 137.5);
2077      g2.draw(lE2D1_2);
2078
2079      // A2D5
2080      if (produktL == 5)
2081        g2.setPaint(c1);
2082      else
2083        g2.setPaint(cStd);
2084
2085      A2D2_1 = new Arc2D.Double(300,185,150,25,0,-180,Arc2D.OPEN);
2086      g2.draw(A2D2_1);
2087      A2D2_2 = new Arc2D.Double(300,160,150,25,0,360,Arc2D.OPEN);
2088      g2.draw(A2D2_2);
2089      Line2D lE2D2_1 = new Line2D.Double(300, 172.5, 300, 197.5);
2090      g2.draw(lE2D2_1);
2091      Line2D lE2D2_2 = new Line2D.Double(450, 172.5, 450, 197.5);
2092      g2.draw(lE2D2_2);
2093
2094      // A2D4
2095      if (produktL == 4)
2096        g2.setPaint(c1);
2097      else
2098        g2.setPaint(cStd);
2099
2100      A2D3_1 = new Arc2D.Double(300,245,150,25,0,-180,Arc2D.OPEN);
2101      g2.draw(A2D3_1);
2102      A2D3_2 = new Arc2D.Double(300,220,150,25,0,360,Arc2D.OPEN);
2103      g2.draw(A2D3_2);
2104      Line2D lE2D3_1 = new Line2D.Double(300, 232.5, 300, 257.5);
2105      g2.draw(lE2D3_1);
2106      Line2D lE2D3_2 = new Line2D.Double(450, 232.5, 450, 257.5);
2107      g2.draw(lE2D3_2);
2108
```

```
2109      // A2D3
2110      if (produktL == 3 || prosess == 5 || knappnr > 0)
2111        g2.setPaint(c1);
2112      else
2113        g2.setPaint(cStd);
2114
2115      A2D4_1 = new Arc2D.Double(300,305,150,25,0,-180,Arc2D.OPEN);
2116      g2.draw(A2D4_1);
2117      A2D4_2 = new Arc2D.Double(300,280,150,25,0,360,Arc2D.OPEN);
2118      g2.draw(A2D4_2);
2119      Line2D lE2D4_1 = new Line2D.Double(300, 292.5, 300, 317.5);
2120      g2.draw(lE2D4_1);
2121      Line2D lE2D4_2 = new Line2D.Double(450, 292.5, 450, 317.5);
2122      g2.draw(lE2D4_2);
2123
2124      // A2D2
2125      if (produktL == 2)
2126        g2.setPaint(c1);
2127      else
2128        g2.setPaint(cStd);
2129
2130      A2D5_1 = new Arc2D.Double(300,365,150,25,0,-180,Arc2D.OPEN);
2131      g2.draw(A2D5_1);
2132      A2D5_2 = new Arc2D.Double(300,340,150,25,0,360,Arc2D.OPEN);
2133      g2.draw(A2D5_2);
2134      Line2D lE2D5_1 = new Line2D.Double(300, 352.5, 300, 377.5);
2135      g2.draw(lE2D5_1);
2136      Line2D lE2D5_2 = new Line2D.Double(450, 352.5, 450, 377.5);
2137      g2.draw(lE2D5_2);
2138
2139      // A2D1
2140      if (produktL == 1)
2141        g2.setPaint(c1);
2142      else
2143        g2.setPaint(cStd);
2144
2145      A2D6_1 = new Arc2D.Double(300,425,150,25,0,-180,Arc2D.OPEN);
2146      g2.draw(A2D6_1);
2147      A2D6_2 = new Arc2D.Double(300,400,150,25,0,360,Arc2D.OPEN);
2148      g2.draw(A2D6_2);
2149      Line2D lE2D6_1 = new Line2D.Double(300, 412.5, 300, 437.5);
2150      g2.draw(lE2D6_1);
2151      Line2D lE2D6_2 = new Line2D.Double(450, 412.5, 450, 437.5);
2152      g2.draw(lE2D6_2);
2153
2154      // Tegner prosess-view
2155      g2.setStroke(bs3);
2156
2157      g2.setPaint(cProsess);
2158      prosessBkg = new RoundRectangle2D.Double(15, 80, 220, 675, 10, 10);
2159      g2.draw(prosessBkg);
2160
2161      // R2D9
2162      if (prosess == 9)
2163        g2.setPaint(c1);
2164      else
2165        g2.setPaint(cStd);
2166
2167      R2D9 = new RoundRectangle2D.Double(25,100,200,50,10,10);
2168      g2.draw(R2D9);
2169
2170      // R2D8
```





```
2171      if (prosess == 8)
2172        g2.setPaint(c1);
2173      else
2174        g2.setPaint(cStd);
2175
2176      R2D8 = new RoundRectangle2D.Double(25,175,200,50,10,10);
2177      g2.draw(R2D8);
2178
2179      // R2D7
2180      if (prosess == 7)
2181        g2.setPaint(c1);
2182      else
2183        g2.setPaint(cStd);
2184
2185      R2D7 = new RoundRectangle2D.Double(25,250,200,50,10,10);
2186      g2.draw(R2D7);
2187
2188      // R2D6
2189      if (prosess == 6)
2190        g2.setPaint(c1);
2191      else
2192        g2.setPaint(cStd);
2193
2194      R2D6 = new RoundRectangle2D.Double(25,325,200,50,10,10);
2195      g2.draw(R2D6);
2196
2197      // R2D5
2198      if (prosess == 5)
2199        g2.setPaint(c1);
2200      else
2201        g2.setPaint(cStd);
2202
2203      R2D5 = new RoundRectangle2D.Double(25,400,200,50,10,10);
2204      g2.draw(R2D5);
2205
2206      // R2D4
2207      if (prosess == 4)
2208        g2.setPaint(c1);
2209      else
2210        g2.setPaint(cStd);
2211
2212      R2D4 = new RoundRectangle2D.Double(25,475,200,50,10,10);
2213      g2.draw(R2D4);
2214
2215      // R2D3
2216      if (prosess == 3)
2217        g2.setPaint(c1);
2218      else
2219        g2.setPaint(cStd);
2220
2221      R2D3 = new RoundRectangle2D.Double(25,550,200,50,10,10);
2222      g2.draw(R2D3);
2223
2224      // R2D2
2225      if (prosess == 2)
2226        g2.setPaint(c1);
2227      else
2228        g2.setPaint(cStd);
2229
2230      R2D2 = new RoundRectangle2D.Double(25,625,200,50,10,10);
2231      g2.draw(R2D2);
2232
```

```
2233      // R2D1
2234      if (prosess == 1)
2235        g2.setPaint(c1);
2236      else
2237        g2.setPaint(cStd);
2238
2239      R2D1 = new RoundRectangle2D.Double(25,700,200,50,10,10);
2240      g2.draw(R2D1);
2241
2242      // knapper
2243      R2knapp01 = new RoundRectangle2D.Double(950,32,50,25,10,10);
2244      g2.draw(R2knapp01);
2245
2246      R2knapp02 = new RoundRectangle2D.Double(950,62,50,25,10,10);
2247      g2.draw(R2knapp02);
2248
2249      // Tegner produktnivå-view
2250      g2.setStroke(bs3);
2251
2252      E = new Ellipse2D.Double(550,75,400,400);
2253      if (produktL > 0 || knappnr > 0)
2254        g2.setPaint(c1);
2255      else
2256        g2.setPaint(cStd);
2257
2258      g2.draw(E);
2259
2260      // Diverse linjer
2261      g2.setStroke(bs0);
2262
2263      g2.setPaint(cStd);
2264      Line2D tH = new Line2D.Double(250, 475, 250, 750);
2265      g2.draw(tH);
2266      Line2D tV = new Line2D.Double(250, 475, 500, 475);
2267      g2.draw(tV);
2268      Line2D tD = new Line2D.Double(500, 475, 530, 400);
2269      g2.draw(tD);
2270
2271      Line2D tArrowH = new Line2D.Double(262.5, 150, 262.5, 400);
2272      g2.draw(tArrowH);
2273      Line2D tArrow_1 = new Line2D.Double(262.5, 150, 250, 175);
2274      g2.draw(tArrow_1);
2275      Line2D tArrow_2 = new Line2D.Double(262.5, 150, 275, 175);
2276      g2.draw(tArrow_2);
2277
2278    }
2279
2280    // Prosedyrer for mushandlinger
2281    public void mouseClicked(MouseEvent me) {
2282      int pX = me.getX();
2283      int pY = me.getY();
2284      System.out.println("meX: " + pX);
2285      System.out.println("meY: " + pY);
2286      produkt(pX, pY);
2287      produktN(pX, pY);
2288      prosess(pX, pY);
2289      knapp(pX, pY);
2290
2291    }
2292
2293    public void mousePressed(MouseEvent me) {}
2294    public void mouseReleased(MouseEvent me) {}
```





```
2295    public void mouseMoved(MouseEvent me) {}
2296    public void mouseDragged(MouseEvent me) {}
2297    public void mouseEntered(MouseEvent me) {}
2298    public void mouseExited(MouseEvent me) {}
2299
2300    // Prosedyrer ved produktvalg
2301    public void produkt(int punktX, int punktY) {
2302
2303      // P01
2304      if (punktY >= 85 && punktY <= 135 && punktX >= 725 && punktX <= 775) {
2305        E1 = 625; E2 = 150; El = 250; produkt = 1;
2306        repaint();
2307      }
2308
2309      // P02
2310      if (punktY >= 100 && punktY <= 150 && punktX >= 790 && punktX <= 840) {
2311        E1 = 625; E2 = 150; El = 250; produkt = 2;
2312        repaint();
2313      }
2314
2315      // P03
2316      if (punktY >= 140 && punktY <= 190 && punktX >= 840 && punktX <= 890) {
2317        E1 = 625; E2 = 150; El = 250; produkt = 3;
2318        repaint();
2319      }
2320
2321      // P04
2322      if (punktY >= 190 && punktY <= 240 && punktX >= 875 && punktX <= 925) {
2323        E1 = 625; E2 = 150; El = 250; produkt = 4;
2324        repaint();
2325      }
2326
2327      // P05
2328      if (punktY >= 250 && punktY <= 300 && punktX >= 890 && punktX <= 940) {
2329        E1 = 625; E2 = 150; El = 250; produkt = 5;
2330        repaint();
2331      }
2332
2333      // P06
2334      if (punktY >= 310 && punktY <= 360 && punktX >= 875 && punktX <= 925) {
2335        E1 = 625; E2 = 150; El = 250; produkt = 6;
2336        repaint();
2337      }
2338
2339      // P07
2340      if (punktY >= 360 && punktY <= 410 && punktX >= 840 && punktX <= 890) {
2341        E1 = 625; E2 = 150; El = 250; produkt = 7;
2342        repaint();
2343      }
2344
2345      // P08
2346      if (punktY >= 400 && punktY <= 450 && punktX >= 790 && punktX <= 840) {
2347        E1 = 625; E2 = 150; El = 250; produkt = 8;
2348        repaint();
2349      }
2350
2351      // P09
2352      if (punktY >= 415 && punktY <= 465 && punktX >= 725 && punktX <= 775) {
2353        E1 = 625; E2 = 150; El = 250; produkt = 9;
2354        repaint();
2355      }
2356
```

```
2357      // P10
2358      if (punktY >= 400 && punktY <= 450 && punktX >= 660 && punktX <= 710) {
2359        E1 = 625; E2 = 150; El = 250; produkt = 10;
2360        repaint();
2361      }
2362
2363      // P11
2364      if (punktY >= 360 && punktY <= 410 && punktX >= 610 && punktX <= 660) {
2365        E1 = 625; E2 = 150; El = 250; produkt = 11;
2366        repaint();
2367      }
2368
2369      // P12
2370      if (punktY >= 310 && punktY <= 360 && punktX >= 575 && punktX <= 625) {
2371        E1 = 625; E2 = 150; El = 250; produkt = 12;
2372        repaint();
2373      }
2374
2375      // P13
2376      if (punktY >= 250 && punktY <= 300 && punktX >= 560 && punktX <= 610) {
2377        E1 = 625; E2 = 150; El = 250; produkt = 13;
2378        repaint();
2379      }
2380
2381      // P14
2382      if (punktY >= 190 && punktY <= 240 && punktX >= 575 && punktX <= 625) {
2383        E1 = 625; E2 = 150; El = 250; produkt = 14;
2384        repaint();
2385      }
2386
2387      // P15
2388      if (punktY >= 140 && punktY <= 190 && punktX >= 610 && punktX <= 660) {
2389        E1 = 625; E2 = 150; El = 250; produkt = 15;
2390        repaint();
2391      }
2392
2393      // P16
2394      if (punktY >= 100 && punktY <= 150 && punktX >= 660 && punktX <= 710) {
2395        E1 = 625; E2 = 150; El = 250; produkt = 16;
2396        repaint();
2397      }
2398
2399    }
2400
2401    // Prosedyrer ved produktnivåvalg
2402    public void produktN(int punktX, int punktY) {
2403
2404      // A2D6
2405      if (punktY >= 100 && punktY <= 150 && punktX >= 300 && punktX <= 450) {
2406        produktL = 6; produkt = 0; prosess = 0; knappnr = 0;
2407        repaint();
2408      }
2409
2410      // A2D5
2411      if (punktY >= 160 && punktY <= 210 && punktX >= 300 && punktX <= 450) {
2412        produktL = 5; produkt = 0; prosess = 0; knappnr = 0;
2413        repaint();
2414      }
2415
2416      // A2D4
2417      if (punktY >= 220 && punktY <= 270 && punktX >= 300 && punktX <= 450) {
2418        produktL = 4; produkt = 0; prosess = 0; knappnr = 0;
```





```
2419        repaint();
2420      }
2421
2422      // A2D3
2423      if (punktY >= 280 && punktY <= 330 && punktX >= 300 && punktX <= 450) {
2424        produktL = 3; produkt = 0; prosess = 6; knappnr = 0;
2425        repaint();
2426      }
2427
2428      // A2D2
2429      if (punktY >= 340 && punktY <= 390 && punktX >= 300 && punktX <= 450) {
2430        produktL = 2; produkt = 0; prosess = 4; knappnr = 0;
2431        repaint();
2432      }
2433
2434      // A2D1
2435      if (punktY >= 400 && punktY <= 450 && punktX >= 300 && punktX <= 450) {
2436        produktL = 1; produkt = 0; prosess = 0; knappnr = 0;
2437        repaint();
2438      }
2439
2440    }
2441
2442    // Prosedyre ved prosessvalg
2443    public void prosess(int punktX, int punktY) {
2444
2445      // R2D9
2446      if (punktY >= 100 && punktY <= 150 && punktX >= 25 && punktX <= 225) {
2447        prosess = 9; produkt = 0; produktL = 0; knappnr = 0;
2448        repaint();
2449      }
2450
2451      // R2D8
2452      if (punktY >= 175 && punktY <= 225 && punktX >= 25 && punktX <= 225) {
2453        prosess = 8; produkt = 0; produktL = 0; knappnr = 0;
2454        repaint();
2455      }
2456
2457      // R2D7
2458      if (punktY >= 250 && punktY <= 300 && punktX >= 25 && punktX <= 225) {
2459        prosess = 7; produkt = 0; produktL = 0; knappnr = 0;
2460        repaint();
2461      }
2462
2463      // R2D6
2464      if (punktY >= 325 && punktY <= 375 && punktX >= 25 && punktX <= 225) {
2465        prosess = 6; produkt = 0; produktL = 3; knappnr = 0;
2466        repaint();
2467      }
2468
2469      // R2D5
2470      if (punktY >= 400 && punktY <= 450 && punktX >= 25 && punktX <= 225) {
2471        prosess = 5; produkt = 0; produktL = 2; knappnr = 0;
2472        E1 = 625; E2 = 150; El = 250;
2473        repaint();
2474      }
2475
2476      // R2D4
2477      if (punktY >= 475 && punktY <= 525 && punktX >= 25 && punktX <= 225) {
2478        prosess = 4; produkt = 0; produktL = 0; knappnr = 0;
2479        repaint();
2480      }
```

```
2481
2482      // R2D3
2483      if (punktY >= 550 && punktY <= 600 && punktX >= 25 && punktX <= 225) {
2484        prosess = 3; produkt = 0; produktL = 0; knappnr = 0;
2485        repaint();
2486      }
2487
2488      // R2D2
2489      if (punktY >= 625 && punktY <= 675 && punktX >= 25 && punktX <= 225) {
2490        prosess = 2; produkt = 0; produktL = 0; knappnr = 0;
2491        repaint();
2492      }
2493
2494      // R2D1
2495      if (punktY >= 700 && punktY <= 750 && punktX >= 25 && punktX <= 225) {
2496        prosess = 1; produkt = 0; produktL = 0; knappnr = 0;
2497        repaint();
2498      }
2499
2500    }
2501
2502    // Prosedyrer ved knappevalgvalg
2503    public void knapp(int punktX, int punktY) {
2504
2505      // Bilde P01
2506      if (punktY >= 32 && punktY <= 57 && punktX >= 950 && punktX <= 1000) {
2507        knappnr = 1; produktL = 0; produkt = 0; prosess = 6;
2508        repaint();
2509      }
2510
2511      if (punktY >= 62 && punktY <= 87 && punktX >= 950 && punktX <= 1000) {
2512        knappnr = 2; produktL = 0; produkt = 0; prosess = 6;
2513        repaint();
2514      }
2515
2516    }
2517
2518  }
```

```
 1 /***
 2 Program: Applikasjonsramme
 3 - Lager en applikasjonsramme
 4   for prototypen og plasserer
 5   denne rammen midt på skjermen
 6
 7 Kildekoden til denne delen er hentet fra:
 8   Java 2D Graphics,
 9   Jonathan Knudsen,
10   O'Reilly, Mai 1999,
11   ISBN: 1-56592-484-3
12 ***/
13
14 import java.awt.*;
15 import java.awt.event.*;
16 import javax.swing.*;
17
18 public class ApplicationFrame
19     extends Frame {
20   public ApplicationFrame() { this("ApplicationFrame v1.0"); }
21
22   public ApplicationFrame(String title) {
23     super(title);
24     createUI();
25   }
26
27   protected void createUI() {
28     setSize(500, 400);
29     center();
30
31     addWindowListener(new WindowAdapter() {
32       public void windowClosing(WindowEvent e) {
33         dispose();
34         System.exit(0);
35       }
36     });
37   }
38
39   public void center() {
40     Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
41     Dimension frameSize = getSize();
42     int x = (screenSize.width - frameSize.width) / 2;
43     int y = (screenSize.height - frameSize.height) / 2;
44     setLocation(x, y);
45   }
46 }
```