Avdelningen för datavetenskap

Tobias Eriksson och Agni Rizk

Grafisk visualisering av en spårbarhetslösning

Graphical visualization of a traceability solution

Datavetenskap
C-uppsats

Datum/Termin:	11-06-09
Handledare:	Donald F. Ross
Examinator:	Stefan Lindskog
Löpnummer:	2011:06

Karlstads universitet 651 88 Karlstad
Tfn 054-700 10 00 Fax 054-700 14 60
Information@kau.se www.kau.se

Grafisk visualisering av en spårbarhetslösning

Tobias Eriksson & Agni Rizk

Denna rapport är skriven som en del av det arbete som krävs för att erhålla en kandidatexamen i datavetenskap. Allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Tobias Eriksson

Agni Rizk

Godkänd, 2011-06-09

Handledare: Donald F. Ross

Examinator: Stefan Lindskog

Sammanfattning

Spårbarhet är processen att följa en specifikation eller produkt från sitt ursprung till sitt slutförande. Att kunna spåra tillverkningen av material eller försäljningen av mat är viktigt och i vissa fall obligatoriskt. Är tillverkningen av en produkt väldokumenterad blir spårningen förhållandevis enkel. Produkten följs i bakvänd ordning; från det att den var klar och till det att den bryts upp i mindre beståndsdelar (till sitt ursprung). Med hjälp av väldokumenterat underlag, databaser och applikationer kan spårningen göras effektiv. Med hjälp av en grafisk visualisering av tillverkningsprocessen blir spårningen mer levande och därmed trevligare.

Konsultföretaget Sogeti som specialiserar sig på lokala IT-tjänster har sett över hur de kan effektivisera spårningen för en av deras befintliga kunder. Det finns redan ett system för spårning hos denna kund i dagsläget, men Sogeti vill titta närmare in på hur de kan göra spårningen effektivare och användarvänligare. De hade en idé om att göra en applikation som presenterar denna spårning visuellt. Dels kan denna specifika kund ha användning av applikationen, dels eventuellt andra kunder.

Under projektets gång har det studerats hur man visualiserar en tillverkningsprocess med avseende på spårningsfunktioner, samt hur man gör den mer tilltalande och användarvänlig. Projektet resulterade i en applikation som möter dessa kriterier.

Graphical visualization of a traceability solution

Abstract

Traceability is the process of following a specification or a product both backwards and forwards from its origin to the completion. Being able to track the production of materials or the sale of food is essential and in some cases mandatory. If the manufacturing process of a product is well documented, tracking becomes relatively simple. It follows the product in reverse order; from the point where the product is completed and breaks it down into smaller components (i.e. to its origin). With the help of good documentation, databases and applications, tracking could be more effective. With the help of a graphical visualization of the manufacturing process, the tracking will be more vivid and thus more pleasant.

The consultant company Sogeti that specializes in local IT services has reviewed how to streamline the tracking on behalf of one of their current customers. The customer already has a system for tracking at present, but Sogeti would like to investigate how to make tracking more efficient and user friendly. They had an idea to make an application that presents the tracking visually. This particular customer as well as other customers could benefit from the use of this application.

During the project, studies of how to visualize manufacturing process (with regard to tracking functions), how to make it attractive and user friendly, were made. The project resulted in an application that meets these criteria.

Tillkännagivanden

Vi vill tacka vår handledare Thomas Heder på Sogeti för hans hjälp och stöd under projektets gång.

Tack även till Åsa Maspers och Bengt Lövenhamn som gav oss möjligheten att få göra detta arbete hos Sogeti.

Vi vill även tacka vår handledare Donald F. Ross på Karlstads universitet för bra stöd och rådgivning.

Slutligen vill vi även tacka John Sören Pettersson, Erik Wästlund och Julio Angulo från informatikavdelningen vid Karlstads universitet för många värdefulla tips och råd.

Innehållsförteckning

Figurförteckning

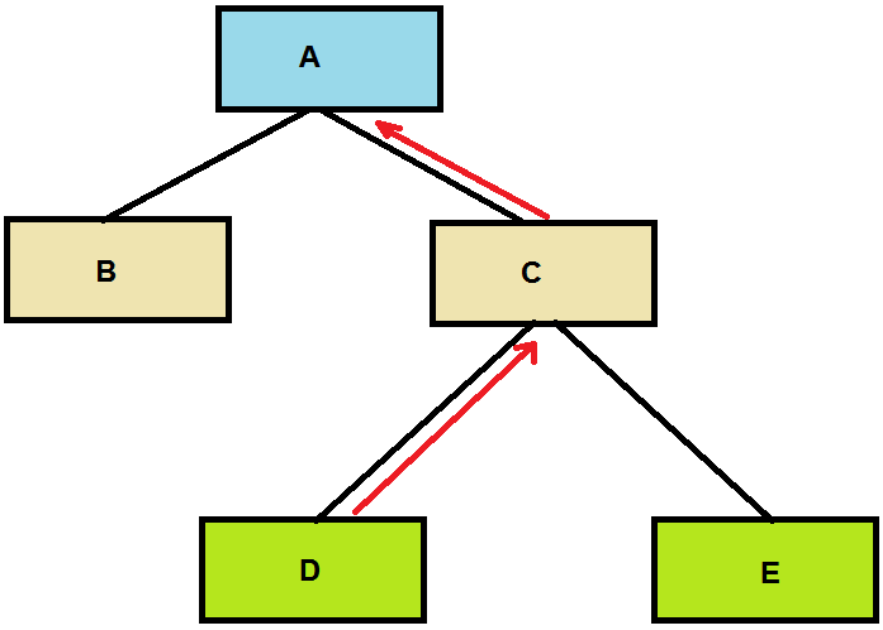
Terminologilista

Analogi	En analogi är en likhet, överensstämmelse eller motsvarighet (i något avseende) [1].
Aluminiumprofil	Aluminiumprofiler framställs genom att aluminium pressas i olika former i stora pressar. Det används bland annat som byggnadsmaterial, inom fordonsindustrin m.m. [2].
Göt	Ett göt är ett metallblock som erhålls då smält metall gjutits i metallformar och där stelnat.
Informationsöverbelastning	För mycket information. Den drabbade kan ha svårt att finna den relevanta informationen då det kan vara svårt att överblicka, välja ut och förstå all information som finns tillgänglig [3]
Komplementfärg	Komplementfärger är av motsatt värde jämfört med primära färger. Om du skulle kombinera en färg med dess komplementfärg skulle resultatet bli vitt [4].
Konsistens	Med konsistent användargränssnitt menar vi i den här uppsatsen att användandet av gränssnittet ska kännas logiskt. Det vill säga att man bör sträva efter till exempel ett konsekvent språkbruk, gruppera liknande funktioner tillsammans (till exempel knappar). Även färger bör användas konsekvent, objekt av samma typ bör alla färgläggas likadant.
Mnemonik	Läran om lämpligaste sättet att utveckla minnet och genom avsiktlig bildade idéassociationer underlätta inlärandet och bevarandet av minneskunskaper [5].
Sackad	Blixtsnabba precisa konjugerade blickriktningsförändringar [6].
Smälta	En mängd av ett ämne som värmts upp kraftigt tills det har smält och som måste hållas uppvärmt för att inte stelna igen

1 Introduktion

Spårbarhet innebär att det är möjligt att följa ett krav, produkt eller dylikt både bakåt och framåt i en kravspecifikation eller produktutveckling [9]. Det här examensarbetet handlar om en applikation som skapar en spårbarhetsgraf

Sogeti är ett konsultföretag som specialiserar sig på lokala IT-tjänster. De har en omfattande verksamhetskunskap inom många olika branscher. En av dessa är inom tillverkandeindustrin [7]. Sogetis kontor i Karlstad utför för tillfället ett arbete för en kund verksam inom aluminiumprofilering. Sogeti har anförtrott oss med ett uppdrag åt denna kund. Uppdraget är att, med hjälp av lagrad spårdata, skapa en applikation som skapar en spårbarhetsgraf av kundens tillverkningsprocess för de olika aluminiumprofilerna de tillverkar. I Figur 1.1 visas ett exempel på en sådan graf. Med hjälp av en graf ska de olika stegen i tillverkningsprocessen ritas ut i kronologisk ordning och syftet är att kunden sedan ska kunna spåra hur ett material tillverkades. Målet är att denna graf (som består av noder och pilar) ska användas som en spårbarhetsgraf. I Figur 1.1 visas ett exempel på en spårning av noden märkt D.



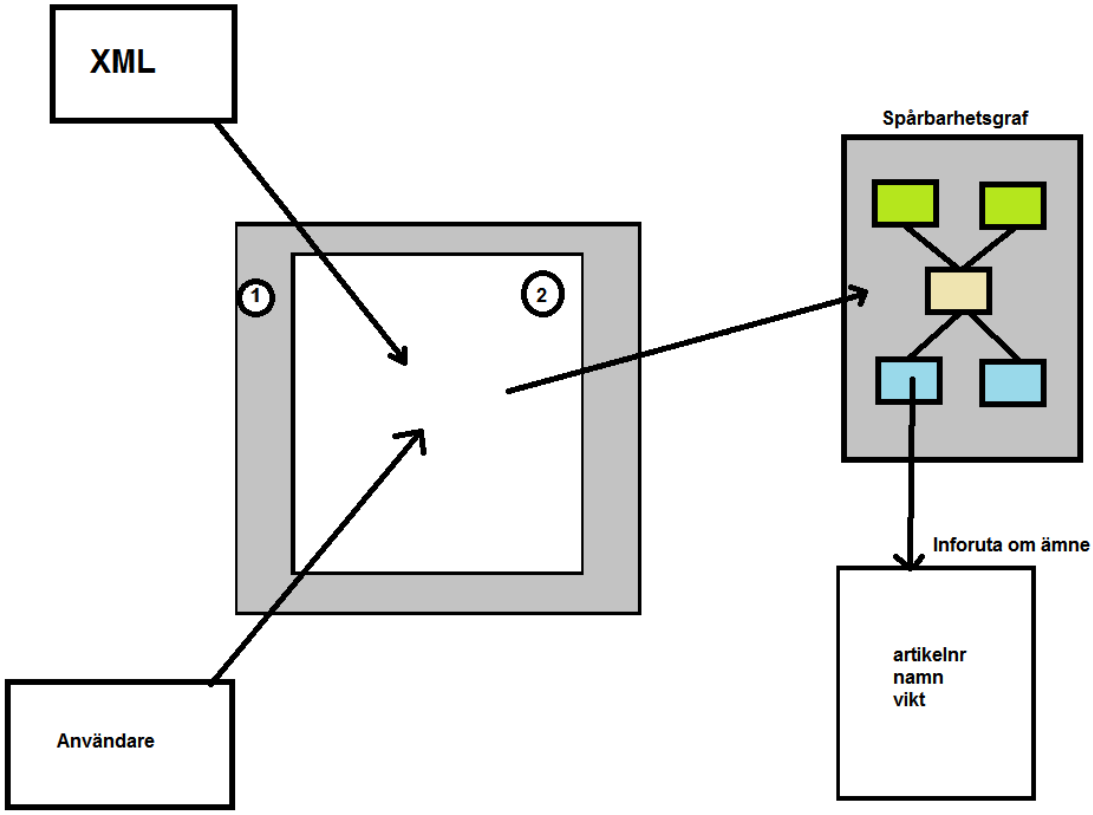
Figur 1.1 Spårning i grafen från D via C till A

Utöver att skapa en applikation som ritar ut en tillverkningsprocess, så ska även applikationen vara lättanvänd och attraktiv samt kunna användas för andra typer av tillverkningsprocesser. Om en applikation som möter de uppsatta målen lyckas att skapas kan applikationen komma att presenteras för Sogetis kund och eventuellt användas i andra projekt.

1.1 Tekniska krav

Applikationen som skapas med hjälp av en grafisk användarkontroll ska rita ut noder och pilar, vilket illustrerads i Figur 1.1. Användarkontrollen kommer att vara en grafisk komponent med underliggande funktioner, som i sin tur kan användas av en fristående applikation som implementerar denna. Funktionerna i användarkontrollen är de funktioner som ska rita upp grafen. En nod i grafen föreställer ett ämne eller en blandning av ämnen och en pil mellan två noder visar operationen mellan dessa samt deras beroende. När en nod klickas på ska information om nodens egenskaper visas. Denna information kan exempelvis vara; vikt, artikelnummer, sammansättning eller annan, för kunden, viktig information. Detta samt resterande krav delas upp som primära och sekundära krav. Se avsnitt 1.2 för detta.

I Figur 1.2 så ges en översikt över hur programmet kan komma att fungera. En huvudapplikation märkt *1* implementerar användarkontrollen märkt *2*. Genom att skicka in data om en spårbarhetsgraf i exempelvis en XML-fil (se 2.4.3 för mer information om XML), så genererar användarkontrollen en spårbarhetsgraf och presenterar denna i sitt fönsterutrymme. Information om noderna och pilarna lagras i spårbarhetsgrafen direkt i dessa och denna information presenteras om användaren klickar på dem.



Figur 1.2 1. Huvudapplikation som implementerar användarkontrollen. 2. Användarkontrollen som skapar en spårbarhetsgraf med hjälp av inskickad data

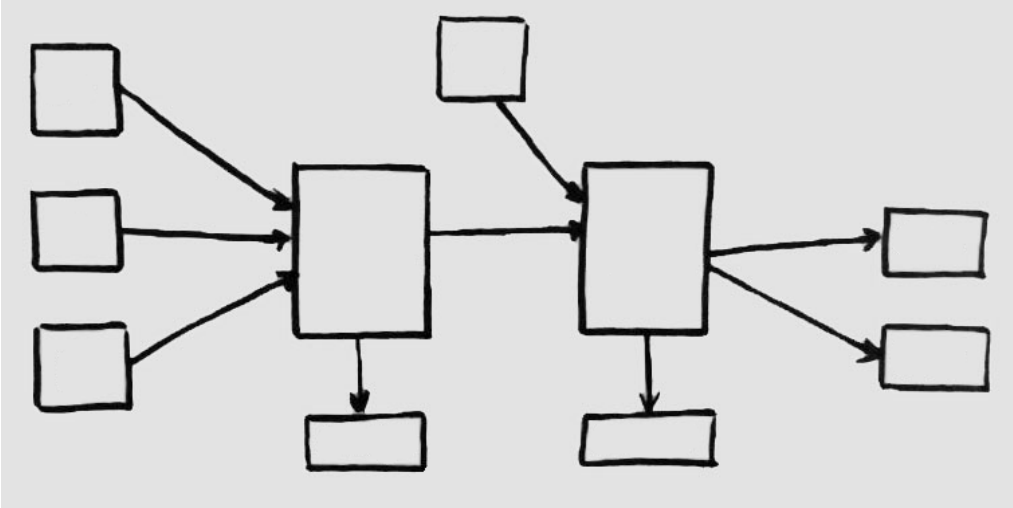
I grafen är det sedan tänkt att användaren ska kunna spåra framåt och bakåt utgående från ett visst ämne.

1.2 Primära krav

För att projektet ska anses vara lyckat är det nödvändigt att nedanstående krav uppfylls.

- Visa en graf med noder och pilar. Figur 1.3 visar en skiss av denna graf.
- Grafens komponenter skall läsas in från en extern källa. Den externa källan kan exempelvis vara en XML-fil som illustrerades i Figur 1.2.
- Utöver informationen som presenteras direkt i noden ska det vara möjligt att presentera ytterligare information om en nods innehåll. Detta kan göras i form av en informationsruta som antingen kan presenteras dynamiskt eller statiskt.

- Användargränssnittet skall hållas enkelt och information ska erhållas kvickt.

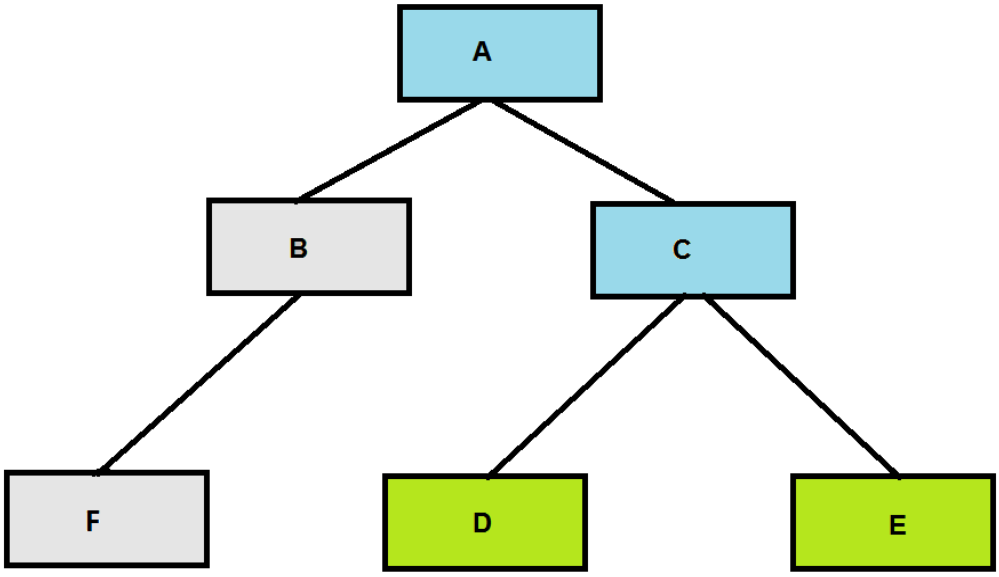


Figur 1.3 Skiss av spårbarhetsgrafen

1.2.1 Sekundära krav

När en graf skapats som uppfyller de primära kraven ska det i mån av tid implementeras följande funktioner.

- Följa ett ämne både framåt och bakåt i grafen.
- Förstora och förminska grafen.
- Spara grafen till en extern källa.
- Presentera två olika ämnens gemensamma ursprung, det vill säga att visa vilka ämnen som de båda har gemensamt bakåt i kedjan. Se Figur 1.4 för ett exempel på hur detta kan se ut. Där visas det gemensamma ursprunget för noderna *D* och *E*.



Figur 1.4 Gemensamma noder för D och E är A och C (färgade blått)

1.3 Icke tekniska krav

Förutom de tekniska kraven är det även ett önskemål att grafen ska ha ett attraktivt användargränssnitt. Följande punkter kommer att studeras och diskuteras för att uppnå detta önskemål.

- Grafens struktur. Hur ska denna graf presenteras?
- Grafiska användargränssnitt. Vad är ett *bra* grafiskt användargränssnitt?
- Grafens användarvänlighet. Hur görs informationssökning användarvänligt?

Dessa frågor diskuteras i denna uppsats och dessa argument kommer i största möjliga utsträckning användas för att göra vårt användargränssnitt attraktivt.

1.4 Kapitelöversikt

Spårbarhetsgrafens funktioner och design är fokuspunkten i denna uppsats. Nedanstående punktlista beskriver vilka områden uppsatsens kapitel berör.

- Kapitel 2 kommer att beröra examensarbetets historia. Hur idén uppkom, varför det är viktigt med spårbarhet, vad som ska utföras och vilka hjälpmedel som finns till förfogande.
- Kapitel 3 behandlar grafiska användargränssnitt. Exempel på grafiska användargränssnitt kommer att ges. Ögonrörelser, färger och informationsbelastning kommer att diskuteras.
- Kapitel 4 består av ett antal prototyper samt diskussioner kring dessa. Ett par lösningsförslag på de olika kraven kommer att demonstreras samt en diskussion angående deras fördelar och nackdelar.
- Kapitel 5 behandlar inläsningen av information om grafens komponenter från en extern källa. I det här fallet inläsning från fil av typen Extensible Markup Language (XML), se avsnitt 2.4.5 för mer information om XML.
- Kapitel 6 åskådliggör examensarbetets resultat och presenterar en utvärdering. Här presenteras även hur grafverktyget och dess funktioner används.
- Kapitel 7 knyter ihop examensarbetet. Lärda erfarenheter, uppfyllda mål och eventuell vidareutveckling diskuteras.

2 Bakgrund

Det här kapitlet kommer att behandla bakgrunden till och syftet med vårt arbete. Avsnitt 2.1 kommer att introducera hur examensarbetet uppstod, studera hur det befintliga spårbarhetssystemet fungerar i dagsläget och fördjupa oss i den del av systemet som ska behandlas. Dessutom kommer det att presenteras förslag till hur en spårbarhetsgraf med tilltalande grafisk design kan skapas. I avsnitt 2.2 diskuteras det varför spårbarhet är viktigt. Avsnitt 2.3 presenterar en kortfattad sammanfattning av den tekniska designen. Avsnitt 2.4 berör de verktyg som används för att nå målet som är att skapa en spårbarhetsgraf. Två befintliga grafritande lösningar presenteras, *Microsoft Automatic Graph Layout* (MSAGL) och *Graph Visualization Software* i avsnitt 2.5. Några korta tankar kring designaspekter kommer att tas upp i avsnitt 2.6. Kapitel 3 kommer att beröra designaspekter djupare.

2.1 Introduktion

Sogeti arbetar med system #1a och #1b, som visas i Figur 2.1. Kundens namn uppges inte i denna uppsats utan kommer i fortsättningen att kalla kunden för *SBH*[1].

Sogeti har anförtrott oss uppgiften att visuellt visa hur tillverkningsprocessen hos SBH går till i System #1a. Det vill säga från det att en order blir mottagen till att materialet blir till ett göt. Se avsnitt 2.1.1 för att se hur långt system #1a sträcker sig.

Det huvudsakliga syftet är att SBH skall kunna använda systemet för att spåra tillverkningen av vissa aluminiumprofiler, utifall en kund har klagomål på någon aluminiumprofil är det viktigt att ta reda på om det eventuellt beror på ett materialfel och lokalisera alla andra kunder

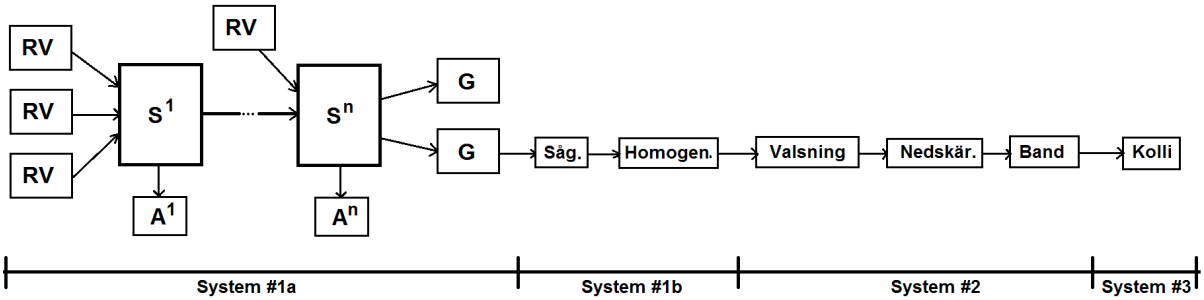
[1] SBH är en akronym för SpårBarHet

som har fått aluminiumprofiler från samma sats/omgång[2]. Dock kommer bara System #1a visualiseras grafiskt och det sträcker sig inte ända till slutkund.

SBH saknar i dagsläget ett bra system för spårbarhet. De har ett par olika system och databaser som de måste gå in i och spårningen kan ta flera timmar om inte dagar. Läs mer om det här i avsnitt 2.1.1.

2.1.1 Nuvarande spårbarhetssystem

För att få ett helhetsperspektiv över SBHs nuvarande spårbarhetssystem så skissade handledaren på Sogeti upp och förklarade kedjan. Figur 2.1 visar vår avbild av denna skiss. Här visas hur långt de tre olika systemen sträcker sig. System #1a behandlar råvarorna, smälter dem och gör dem till göt. System #1b sågar och homogeniserar göten. System #2 behandlar valsning och nedskärning som sedan resulterar i band. System #3 behandlar emballering och kolli-ID samt spårar var det skickas.



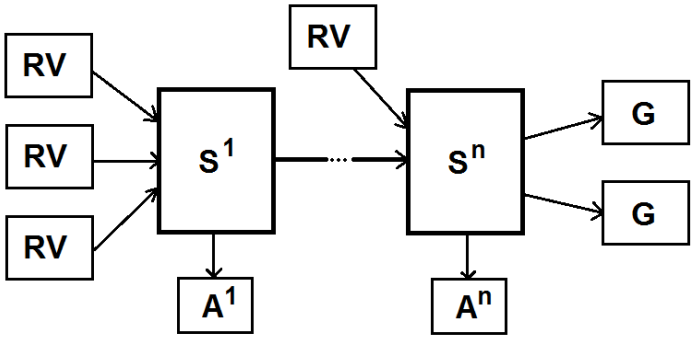
Figur 2.1 Skiss på hela tillverkningsprocessen

Vårt fokus är som sagt endast på System #1a, men då det finns en möjlighet att dessa system faktiskt slås ihop i framtiden så kommer det finnas i åtanke att försöka göra vår applikation anpassningsbar.

[2] Eng. batch

2.1.2 System #1a

När SBH mottagit en order på en viss blandning metaller så börjar de tillverkningen utifrån ett recept. Ett antal råvaror eller grundämnen [8] såsom aluminium, magnesium, zink et cetera blandas ihop i en smälta. Därefter tas det analyser av smältan, resultatet av analysen styr resten av tillverkningsprocessen. Analysen kan skickas liksom resten av komponenterna in i användarkontrollen via en XML-fil, se Figur 1.2 för en illustration av detta. Är analysen inte tillfredställande, behöver det tillsättas mer av ett ämne och det sker oftast i samma ugn. Skulle ugnen bli full, kan delar av smältan överföras till en ny ugn. Analysresultatet påverkar hur många gånger råvaror tillsätts. När analysen är tillfredställande görs det göt av smältan. Denna del av tillverkningsprocessen har valts att kallas för System #1a, se Figur 2.2.



Figur 2.2 Skiss på System #1a av tillverkningsprocessen

RV står för en råvara, *S* för smälta, *A* för analys och *G* för göt. När det tillsätts nya råvaror till smältan kan det ses som en ny smälta även om det sker i samma ugn. För att det ska bli lättöverskådligt visas processen som en ny process som beror på den äldre i kronologisk ordning. Då varje smälta har en analys visas analysen som en del i smältan. Därav kommer analysen inte presenteras i en egen nod utan de kommer ses som tilläggsinformation av smältan.

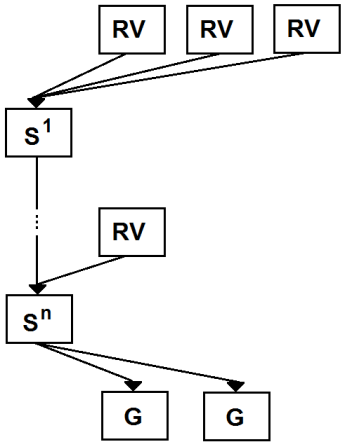
2.1.3 Vårt förslag

Huvudsyftet med detta projekt är att med hjälp av lagrad spårdata summera och presentera tillverkningsprocessen grafiskt. Det enda kravet från Sogeti är att presentera denna process

med minimal funktion, med andra ord utan icke nödvändiga finesser som till exempel ett sökfält för automatiskt sökning av råvaror, smälta och så vidare. Då applikationen ska ha minimalt med funktionalitet och finesser kommer det inte att köpas in ett grafhanteringsverktyg, utan befintliga metoder kommer att undersökas för att eventuellt få idéer på design och lösningar. Om lösningarna är tillfredsställande och kostnadsfria så kommer det eventuellt att bygga vidare på dessa.

Då väldigt fria händer har getts så har följande prioriteringslista tagits fram:

- Presentera grafen uppifrån och ner (vertikalt istället för horisontellt).
- Följa ett ämne både framåt och bakåt i grafen. Det vill säga att presentera dess ursprung och de ämnen som det leder till.
- Färglägga de olika typerna noderna. Till exempel får råvarorna en viss färg, smälta en annan och göt ytterligare en annan färg.
- Kunna zooma in och ut i grafen.
- Skapa ett attraktivt användargränssnitt.
- Funktionalitet för att jämföra analysen av olika smältor.
- Lösningen ska vara kostnadsfri utöver de verktyg som presenteras i avsnitt 2.4.
- Kunna spara och läsa in en graf från en extern fil. Sogeti har som önskemål att använda filer av typen *Extensible Markup Language*, se avsnitt 2.4.5.



Figur 2.3 Skiss på hur tillverkningsprocessen önskas att bli presenterad

Vår högsta prioritering är att representera grafen vertikalt istället för horisontellt. Skillnaden kan ses vid jämförelse mellan Figur 2.2 och Figur 2.3. Detta på grund av att tillverkningsprocessen i teorin kan bli tämligen lång (ett tiotal smältor). Genom att presentera grafen vertikalt ges det med hjälp av nivåindelningen och datormusens rullningshjul bra översikt hur tillverkningen har skett. Färger kommer att väljas med stor omsorg då färgerna inte endast är till för att pryda grafen utan även är till för att förbättra översiktsbilden.

2.2 Spårbarhet

Spårbarhet är viktigt inom industrier såsom fordons- [10], läkemedels- [11] och livsmedelsindustrin [12]. Om det till exempel förekommer något fel på en komponent till ett visst fordon är det viktigt att kunna spåra alla fordon med samma felaktiga komponent och återkalla dem. Detsamma gäller inom läkemedels- och livsmedelsindustrin. Det är viktigt att kunna spåra piller från fabrik till mun och mat från jord till butik.

"The requirements traceability is the ability to describe and follow the life of a requirement, in both a forward and backward direction, i.e. from its origins, through its development and specification, to its subsequent deployment and use, and through periods of ongoing refinement and iteration in any of these phases." [13].

Citatet ovan beskriver spårbarhet väldigt väl. Här talas det om spårbarhet för krav, att det skall vara möjligt att följa dem framåt och bakåt, från sitt ursprung, genom utvecklingen och sedan tillbaka igen. Detsamma gäller exempelvis komponenter i en elektronisk utrustning. Komponenterna behöver spåras från ursprungstillverkaren till hur de löds ihop i den elektroniska utrustningen, till hur de samverkar med andra komponenter till vem som sedan köper denna utrustning och bakåt igen.

2.3 Teknisk design

Slutprodukten kommer inte vara en fristående applikation utan ett återanvändbart verktyg, en så kallad *användarkontroll*, som ska visualisera den data som skickas in i den. Det innebär i

detta fall att andra grafiska *.NET*-applikationer (se avsnitt 2.4.1 för mer information om .NET) kommer att kunna använda denna och skicka in data i den för att visualisera en spårbarhetsgraf. Ett krav som därför ställs på produkten är att den har ett väl definierat *Application Programming Interface* (API) [14] vilket är en regeluppsättning för hur en viss programvara kan kommunicera med annan programvara.

2.4 Verktyg

Samtliga verktyg som presenteras i det här avsnittet är de verktyg som Sogeti önskat ska användas. Ramverket .NET kommer att användas för att utveckla den önskade produkten. Utvecklingen kommer att ske i *Microsoft Visual Studio 2010* [15] och Sogeti har valt att *Windows Presentation Foundation* (WPF) ska användas. Programmet kommer att hämta data från en fil av typen XML. Även detta var ett önskemål från Sogeti.

2.4.1 Ramverket .NET

Ramverket .NET är ett mjukvaruramverk för operativsystemen *Microsoft Windows* [16]. Det har två huvudsakliga komponenter; *Common Language Runtime* (CLR) [17] och klassbiblioteket för .NET [18].

CLR erbjuder en exekveringsmiljö för programkod. Tjänster för bland annat minneshantering, trådhantering, kompilering, typhantering samt andra typer av säkerhetstester för att garantera robusthet tillhandahålls av CLR. Kod som är riktat mot CLR benämns *hanterad kod* och kod som inte har CLR som mål benämns *ohanterad kod* [18].

Klassbiblioteket innehåller en rad återanvändbara typer som är nära integrerade med CLR. Biblioteket är objektorienterat och innehåller typer som i sin tur tillhandahåller ytterligare funktioner. Exempelvis finns typer för stränghantering, datainsamling, databasåtkomst och filåtkomst.

2.4.2 Windows Presentation Foundation

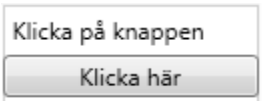
Windows Presentation Foundation (WPF) [19] är ett grafiskt presentationssystem för Microsoft Windows och är designat för .NET. WPF låter utvecklaren utveckla applikationen användandes både märkspråk som bestämmer utseendet på applikationen och bakomliggande kod för att bestämma dess beteende. Märkspråket är vanligtvis *Extensible Application Markup Language* (XAML) [20] och bakomliggande koden skrivs i ett hanterat programspråk, i vårt fall Visual C# [25].

2.4.3 Extensible Application Markup Language

XAML är ett deklarativt märkspråk [21] som primärt används för att instansiera .NET-objekt och främst grafiska användargränssnitt i WPF. Detta innebär bland annat definition av hur paneler, knappar och kontroller m.m. som ska arrangeras i WPF-applikationer. I Figur 2.4 visas ett kodfragment som resulterar i utseendet i Figur 2.5.

```
<StackPanel>
    <Label Content="Klicka på knappen"/>
    <Button Content="Klicka här"/>
</StackPanel>
```

Figur 2.4 Kodexempel WPF



Figur 2.5 Resultat av kod i Figur 2.4

2.4.4 Microsoft Visual Studio 2010

Microsoft Visual Studio 2010 [15] är en samling utvecklingsverktyg från Microsoft som används för att bygga webbapplikationer i *ASP.NET* [22], webbtjänster med *XML* [23] och

applikationer. Det finns stöd för flera programspråk som exempelvis *Visual Basic* [24], *Visual C#* [25] och *Visual C++* [26], som alla använder samma integrerade utvecklingsmiljö. Detta underlättar lösningar där flera programspråk är inblandade.

2.4.5 Extensible Markup Language

Extensible Markup Language (XML) [27] är ett märkspråk vars syfte är att utbyta data mellan olika system. Dess grundsyntax har följande utseende.

```
<namn>Fredrik</namn>
```

Genom att kombinera flera element i en hierarki är det möjligt att beskriva strukturer hos information. Till exempel en telefonlista som visas nedan.

```
<?xml version="1.0" encoding="UTF-8"?>
<telefonlista>
    <person>
        <namn>Adam</namn>
        <telefonnummer>070123456789</telefonnummer>
    </person>
    <person>
        <namn>Eva</namn>
        <telefonnummer>070123443529</telefonnummer>
   </person>
</telefonlista>
```

Den första raden ovan är en så kallad XML-deklaration som beskriver vilken version av XML som används samt vilken teckenkodning som används, i det här fallet UTF-8 [28].

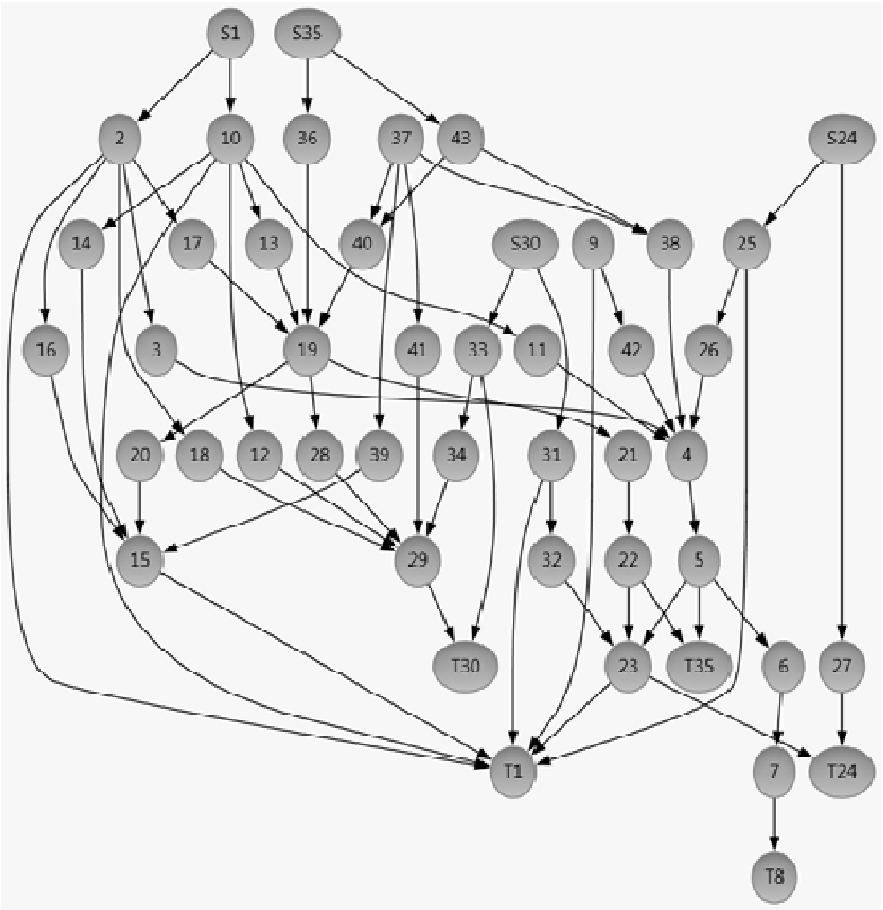
2.5 Grafhanteringsverktyg

Det finns ett antal för oss kända grafhanteringsverktyg som tillhandahåller liknande funktionalitet som den som eftersöks i vårt projekt. Dessa är *Microsoft Automatic Graph Layout* (MSAGL) och *Graph Visualization Software* (Graphviz). Dessa

grafhanteringsprogram har studerats då de föreslogs av Sogeti. Detta avsnitt ger en kortare översikt och diskussion av dessa.

2.5.1 Microsoft Automatic Graph Layout

Microsoft Automatic Graph Layout (MSAGL) [29] är ett verktyg i .NET vars syfte är att automatiskt skapa en graf från inskickad data. Den bygger på principerna för *Sugiyama scheme* som skapar lager- eller hierarkiska layouter. Se Figur 2.6 för ett exempel på en graf som har skapats i MSAGL [30].

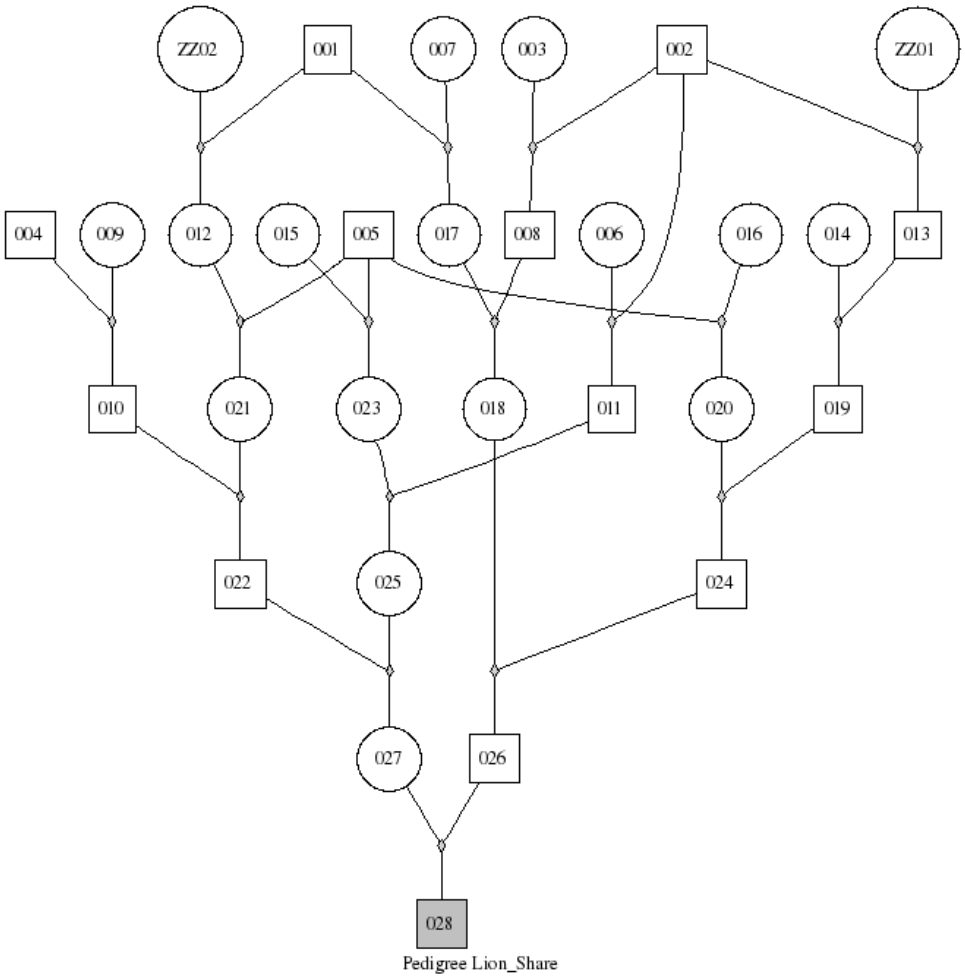


Figur 2.6 Graf skapad i MSAGL

MSAGL kan vara en bra lösning på hur en spårbarhetskedja kan visualiseras men dess nackdel är att verktyget kostar 279,95 amerikanska dollar (den 24 april 2011) i *Microsoft Store* [31].

2.5.2 Graph Visualization Software

Graph Visualization Software (Graphviz) [32] är en öppen programvara för att visualisera grafer. Den tar indata från enkla textspråk och returnerar grafer och diagram i format såsom *PDF*, *PostScript* och ett flertal andra bildformat. Nackdelen med detta är att grafen inte är särskilt interaktiv då den sparas i dessa format. Exempelvis kanske användaren skulle önska att få mer info om denne klickar på en nod i grafen, och då få upp en popup-ruta med utökad information om noden. Detta är inte möjligt i ovan nämnda format. I Figur 2.7 visas en graf som är skapad i Graphviz [33].



Figur 2.7 Graf skapad i Graphviz

2.6 Designaspekter

I ett experiment som detta är det viktigt att tänka på designaspekter. Hur blir en graf lättläst? Vad är ett bra grafiskt användargränssnitt? Det måste studeras vad människor förväntar sig av ett grafiskt användargränssnitt, hur människor läser grafer, vilken struktur grafen bör ha för att ögat ska arbeta minimalt samt vilka färger som bör och inte bör användas. Målsättningen är att hålla designen nedskalad för att på så sätt hålla fokus på själva spårningen. Dessa designaspekter och många fler kommer att diskuteras mer ingående i Kapitel 3.

2.7 Summering

I detta kapitel har det diskuterats bakgrund och syftet med det här examensarbetet. Målet är att utveckla ett program som visualiserar en tillverkningsprocess av göt. Gestaltningen kommer att ske i form av en graf som visar händelserna i kronologisk ordning. Kapitel 2 har diskuterat hur de nuvarande spårbarhetssystemen ser ut i dagsläget samt våra förslag till hur den del av systemet som ska behandlas grafiskt kan representeras. Det har även kort talats om varför spårbarhet är viktigt. En kortare presentation har även getts av de verktyg som kommer att användas. Två program som också ritar ut grafer har studerats och diskuterats kring. Både MSAGL och Graphviz tillhandahåller helt eller delvis den funktionalitet som efterfrågas i vårt projekt men dock har det inte valts att bygga vidare på någon av dessa lösningar. Anledningen till detta är att MSAGL har en licenskostnad vilket står i konflikt med prioriteringen att lösningen ska vara kostnadsfri vilket nämndes i avsnitt 2.1.3. Graphviz faller också bort då det inte skapar tillräckligt interaktiva grafer. Om användaren exempelvis vill att färgen på en nod ska ändras när muspekaren förs över den, är det inte möjligt i Graphviz. Därför väljs det att utveckla en egen lösning (med hjälp av verktygen som presenterades i avsnitt 2.4) som dels är gratis, dels ska vara mer interaktiv. Designaspekter har kort diskuterats och mycket mer om detta kommer i Kapitel 3.

3 Designaspekter

Detta kapitel behandlar och diskuterar designaspekter för grafiska användargränssnitt. I kapitlet definieras vad ett grafiskt användargränssnitt är och vad som betraktas som ett bra sådant. Det diskuteras även för projektet relevanta parametrar som hur ögon fixerar och förflyttas samt hur färger samt informationsbelastning inverkar på användaren. Dessa aspekter skall ligga till grund för hur vår applikation skall utformas.

När det studeras designaspekter, väljs det att granska webbsidor istället för applikationer då dessa är mer lättillgängliga och behöver inte installeras. Designprinciperna är i princip också likadana för webbsidor och applikationer så detta val påverkar inte relevansen för vårt examensarbete.

3.1 Grafiska användargränssnitt

Detta avsnitt behandlar övergripande vad ett grafiskt användargränssnitt är, vad det har för syfte och vilka som använder det. Några riktlinjer för hur bra användargränssnitt skapas kommer att introduceras. Avsnittet kommer även att presentera och diskutera ett exempel på grafiskt användargränssnitt.

3.1.1 Definition

Ett gränssnitt är en utformning av en förbindelse mellan olika objekt [34]. Ett användargränssnitt är en länk mellan en användare och en maskin eller programvara. Användargränssnittet ger möjlighet till två saker.

- Inmatning, det vill säga, det ger användaren möjlighet att påverka systemet.
- Utdata, det vill säga, systemet presenterar information och resultat.

Det kan antingen vara grafiskt eller textbaserat (kommandobaserat) [35]. Ett grafiskt användargränssnitt är ett användargränssnitt som möjliggör en kommunikation mellan

människa och dator. Det grafiska användargränssnittet presenterar information och åtgärder tillgängliga för användaren genom bilder, symboler och grafiska element som utgör metaforer eller analogier till objekt i verkligheten [36, 37].

Så vad är då ett bra användargränssnitt? En definition som utgörs av åtta regler (eller riktlinjer då en sådan lista aldrig kan tillämpas på alla scenarion) benämns härefter *gyllene reglerna*. De presenteras av Schneiderman och Plaisants, och reglerna är som följer [38]:

1. *Sträva efter konsistens*. En av de regler som oftast bryts och att strikt följa den kan vara väldigt svårt då många olika typer av konsistens existerar. Olika former av konstisens kan vara; identiska terminologier skall användas i menyer och hjälpmenyer; färgval, layout, användning av stora och små bokstäver samt teckensnitt.

2. *Sträva efter att tillgodose alla typer av användare*. Identifiera vilken typ av användare som skall använda gränssnittet och anpassa efter den. Ofta finns det flera olika typer av användare med olika erfarenhet, ålder och handikapp. Genom att lägga till tjänster för nybörjare samt tjänster för experter kan gränssnittet förbättras.

3. *Ge informativ återkoppling*. För varje operation en användare utför i gränssnittet ska denne få tillbaka ett informativt svar. För operationer som utförs frekvent och anses vara lite mindre kan det ges en mer subtil återkoppling, större operationer bör dock ge en substantiell återkoppling.

4. D*esigna dialoger som ger avslut.* Sekvenser av åtgärder bör organiseras så att de har en början, en mittendel och ett slut. Användaren bör ges ett tillfredsställande genomförande och slutförande, oavsett om de använder stationära eller mobila enheter. Exempelvis förflyttar e-handelssidor användaren från det att han eller hon lagt sina varor i kundkorgen till kassan som sedan slutar med en tydlig bekräftelsesida som avslutar transaktionen.

5. *Förebygg fel*. Så långt som möjligt bör systemet vara byggt så att användaren inte kan göra allvarliga fel. Exempelvis kan knappar ”gråas ut” när de inte bör användas (som

är olämpliga för tillfället) och att inte tillåta bokstäver i textrutor som endast förväntar sig siffror. Om användaren utför ett fel ska systemet upptäcka det och ge enkla och utförliga beskrivningar hur problemet kan lösas.

6. *Tillhandahåll funktioner för att ångra handlingar*. Så långt det är möjligt ska det strävas efter att kunna göra utförda handlingar av ogjorda. Det ger användaren en trygghet då de vet att eventuella fel han eller hon skapar kan göras ogjorda vilket leder till att han eller hon vågar använda obekanta operationer.

7. *Stödja den interna ”locus of control”*[3] [39]. Erfarna användare önskar känna att de är de som styr gränssnittet och att gränssnittet svarar på deras handlingar. Överraskade gränssnittshandlingar, tråkiga sekvenser av dataposter, oförmåga att få eller svårt att få nödvändig information och oförmåga att få den åtgärd som önskades utförd skapar oro och missnöje.

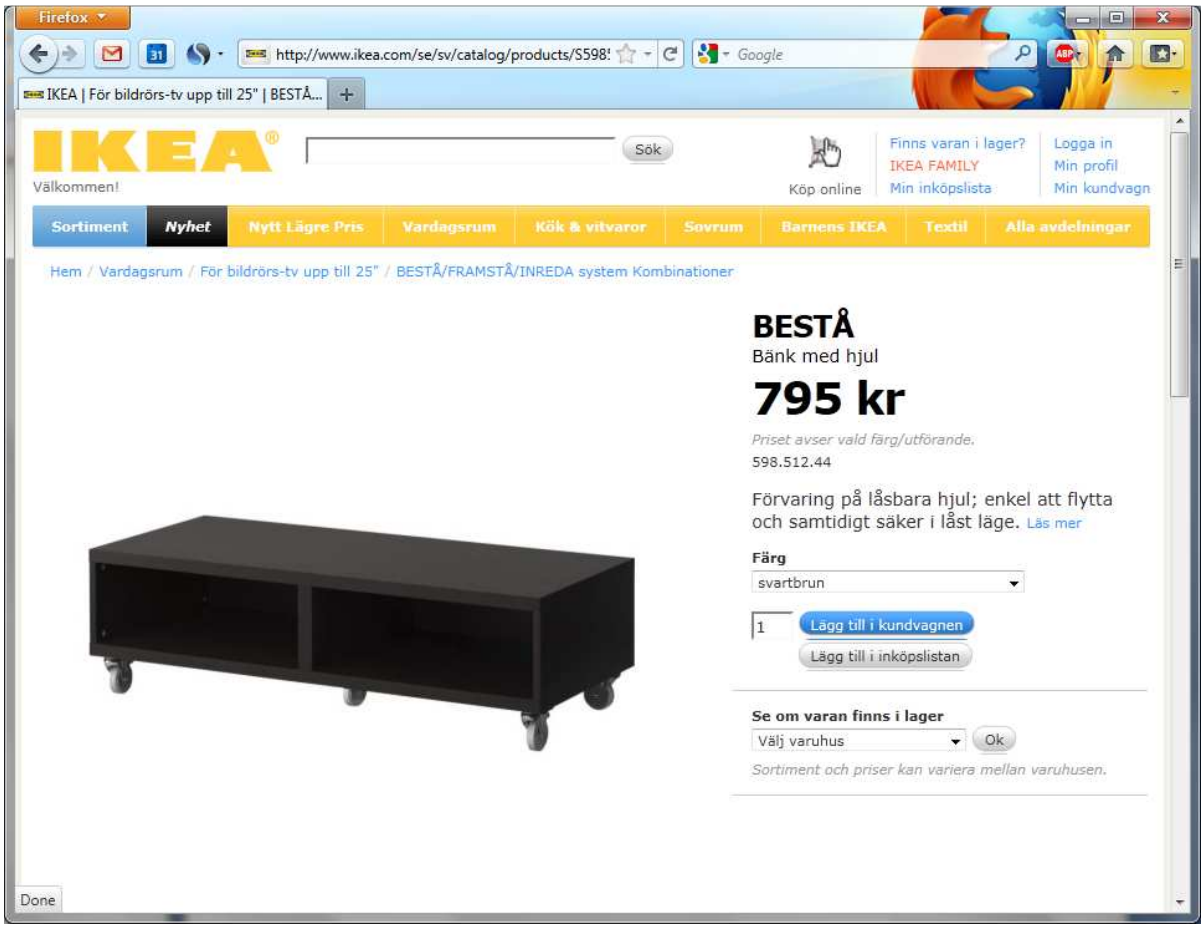
8. *Reducera användningen av korttidsminnet*. Då en människas korttidsminne är begränsat krävs det att användargränssnittet hålls enkelt, att rörelser i fönstret reduceras och att användaren tillåts att bekanta sig med eventuella koder och mnemonik som används i systemet.

3.1.2 Exempel på grafiskt användargränssnitt

I Figur 3.1 visas ett exempel på ett grafiskt användargränssnitt som är möbelföretaget Ikeas hemsida. Där visas många exempel på grafiska element såsom hyperlänkar, rullgardinslistor, bilder och knappar. Färgvalet består i huvudsak av fyra färger som återkommer på hemsidans alla sidor; vit, gul, blå och svart. Detta uppfyller kraven på konsistens för färger som nämns i

[3] ”Locus of control” refererar till den grad människor tror att de kan kontrollera händelser som påverkar dem. Människors ”locus of control” kan antingen vara intern (personen upplever att den kontrollerar sitt liv) eller extern (personen upplever att miljön, någon högre makt eller andra människor styr dennes beslut och liv).

den första av *gyllene reglerna* ovan. Färger kommer att diskuteras mer ingående i avsnitt 3.3. Den övre menyn grupperas liknande produkter som till exempel möbler för vardagsrum, kök och sovrum. Detta uppfyller även en konsistens med avseende på den första regeln.



Figur 3.1 Skärmdump över Ikeas hemsida (http://www.ikea.se) tagen den 2 maj 2011

När en vara läggs till i kundvagnen med hjälp av den blåfärgade knappen i Figur 3.2 ovan så dyker en ruta med återkoppling upp. Att lägga till en vara i kundvagnen kan anses vara en vanlig operation vid e-handel och därför ska inte bekräftelserutan ta för mycket uppmärksamhet (eller täcka en stor procentuell yta av skärmen) utan bara ger en kort bekräftelse på att varan verkligen lades till i kundvagnen. Ikeas återkopplingsruta som visas i Figur 3.2 tar inte mycket uppmärksamhet i anspråk och kräver ingen återkoppling från användaren. Om användaren klickar sig vidare till en annan vara så försvinner rutan utan att göra några ändringar i systemet. Denna funktion uppfyller därför den tredje regeln i de *gyllene*

reglerna där riktlinjen menar att det ska ges återkoppling som ska vara mer eller mindre anspråksfull beroende på om en operation är liten och frekvent eller stor. Dock skulle det önskas att det räckte med att trycka någonstans utanför popup-rutan för att få den att försvinna istället för att trycka på *stäng*. Hemsidan skulle kunna erbjuda båda alternativen, på så sätt skulle nya e-kunder känna sig trygga och erfarna e-kunder som kanske handlar mycket och ofta skulle bara kunna klicka någonstans utanför återkopplingsrutan och på så sätt snabbare återgå till att handla. Detta skulle uppfyllt den andra regeln som förespråkar om att tillgodose alla typer av användare i de *gyllene reglerna*. För att uppfylla regel nummer sex, *tillhandahåll funktioner för att ångra handlingar*, behövs det en ångra knapp i popup-rutan. Då alternativet *Lägg till i kundvagnen* är nära *Lägg till i inköpslistan* kan feltryckningar enkelt uppstå. Det går att klicka på *kundvagnen* och göra handlingen ogjord där men det är relativ omständligt. Det skulle, åter igen, vara möjligt att ge användaren båda möjligheterna.

Figur 3.2 Återkopplingsrutan som visas när en vara läggs till i kundvagnen

En riktlinje som sidan bryter mot är den femte regeln, det vill säga att förebygga fel. Om en bokstav matas in i rutan som anger antal produkter av en viss vara och sedan trycker på *Ok* så lägger systemet till exakt en vara i kundkorgen. Användaren informeras alltså inte att en bokstav har tryckts in istället för siffra och detta skulle kunna leda till situationen att användaren inte får det önskade antalet av en viss vara. Ett exempel är att användaren önskar

sju stycken av varan men missar tangenten *7* på tangentbordet och träffar den närliggande[4] tangenten *U*, om inte användaren själv uppmärksammar detta och lägger en beställning så får den potentiella kunden endast ett styck av denna vara istället för som önskat sju stycken.

Förs muspekaren över den gula menyn i Figur 3.3 så färgas den knappen orange, vilket illustreras i Figur 3.3. Detta tydliggör vilken knapp som är markerad och på så sätt gör sidan lite attraktivare samt att den känns lite mer interaktiv.

Figur 3.3 Färgen på knappen Sovrum färgas orange när musen förs över den

I Figur 3.4 visas en förstorad bild av högra hörnet av Figur 3.1. Där tillhandahålls en funktion som gör det möjligt att se lagerstatus i vald butik för den vara som är vald. Varuhuset väljs med hjälp av rullgardinslistan och därefter kan knappen märkt med *Ok* klickas. Ett stapeldiagram dyker då upp som består av mellan noll till fem block. Ju mer block desto större kvantitet av varan finns i lagret. Staplarna är också färgade gröna för att ge en ökad förståelse att många varor finns i lagret.

[4] För ett tangentbord med en tangentlayout av typen QWERTY

Figur 3.4 Funktion för att se lagerstatus för en viss produkt. 1. Välj stad, 2. Lagerstatus presenteras som stapeldiagram

Klickas den blåfärgade länken *Mer information om lagerstatus* så dyker sidan som visas i Figur 3.5 upp. Denna visar det exakta antalet av den varan som finns i lagret, i det här fallet åtta stycken. Nedan visas en färgindikator med färgerna grön, gul och röd. Om det finns så få av en vara på lager så att varan riskerar att säljas slut inom kort så växlar stapeln till gul. Är varan helt slutsåld så växlas färgen på stapeln till röd. Dessa färger är vanligt förekommande indikatorfärger, men kan kritiseras då den kan försvåra för den röd-gröna färgblindheten som diskuteras närmare i avsnitt 3.3. Dock presenteras det även i textform hur många varor som finns på lager, så färgkodningen bryter inte mot regel nummer två i *gyllene reglerna*.

Figur 3.5 Utökad information om lagerstatus för varan på angivet varuhus

Det har nu diskuterats kring de mest vitala delarna av hemsidan för Ikea. Den uppfyller nästan alla regler av de åtta *gyllene reglerna*. Något som kan kritiseras är att den inte uppfyller regel nummer fem då det är tillåtet att lägga till exempelvis *U* stycken av en vara i kundkorgen.

Det som kan förbättras är reglerna två och sex. Återkommande kunder bör ha *klick och tricks* som gör deras handel effektivare (och roligare). Dessutom bör användaren kunna ångra knapptryckningar direkt.

Resterande regler är väl uppfyllda, användaren har exempelvis alltid den interna känslan av ”locus of control” när Ikeas hemsida besöks. Dialogerna ger alltid ett bra avslut, vare sig användaren lägger till något i kundkorgen, kollar tillgänglighet eller avslutar ett köp.

3.1.3 Relevans till vårt projekt

Till mångt och mycket så har vårt projekt flera gemensamma krav med Ikeas hemsida. Den ska hämta data från en underliggande databas (i vårt fall en XML-fil), den ska bestå av

grafiska komponenter innehållande information om ämnen (eller i Ikeas fall möbler). De åtta *gyllene reglerna*, som diskuterades tidigare, gäller för såväl hemsidor som applikationer.

3.2 Ögonrörelser

När människan läser har det ingen praktisk betydelse i vilken riktning skriften löper. I väst läses från vänster till höger, i stora delar av mellanöstern läses från höger till vänster och i en rad östasiatiska länder läses uppifrån och ned. Faktum är att hjärnan uppfattar tecken lika bra oavsett åt vilket håll skriften löper. Flera försök har visat att det inte krävs mer än några timmars övning, förrän det går lätt att läsa de latinska bokstäverna flytande i motsatt riktning, det vill säga från höger till vänster [40]. Testa på nedanstående mening.

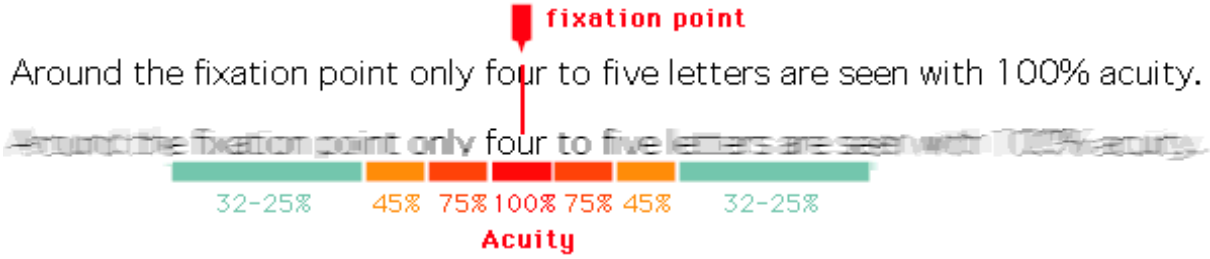
.repöl netfirks gnintkir nekliv i esledyteb ksitkarp negni ted rah resäl nam räN

På grund av att ögonen förflyttar sig på samma sätt vare sig ett stycke text läses eller en bild betraktas, blir detta intressant även för vårt projekt då vår graf kan betraktas som en bild innehållande text. Det spelar alltså ingen roll om en graf presenteras lodrätt eller vågrätt. Även om en graf har betraktats lodrätt under en lång tid och sedan betraktas samma graf vågrätt så tar det inte lång tid att ändra sin mnemonik.

3.2.1 Fixeringar och sackader

Vid läsning av text eller vid en bildbetraktelse så förflyttar sig ögonen på två olika sätt [41]. Antingen fixerar ögonen på en punkt eller så görs en hastig ögonrörelse mellan två punkter, en så kallad sackad[5]. Figur 3.6 illustrerar hur stort område kring en fixering som kan ses med en viss procentuell noggrannhet [42].

[5] Eng. saccad



Figur 3.6 Illustration av fokus runt en fixeringspunkt

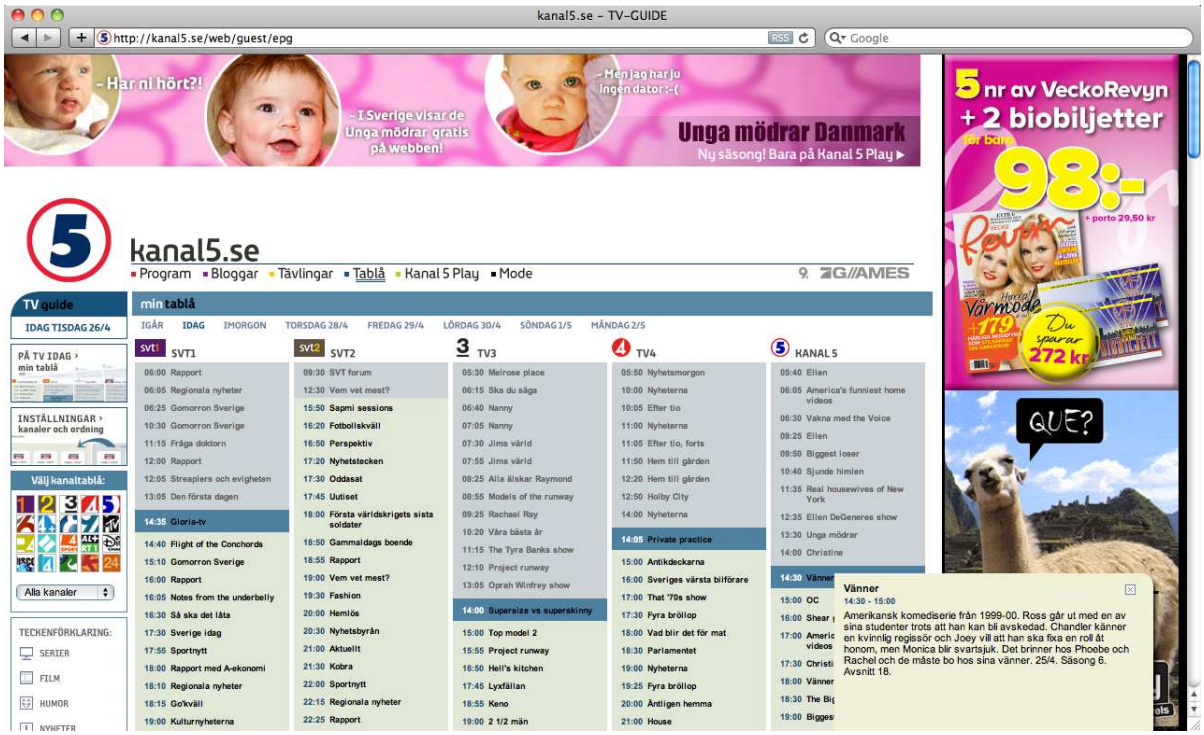
Figuren visar en textrad skriven på engelska. Texten lyder; ”Around the fixation point only four to five letters are seen with 100 % acuity.”. Läsaren har i denna mening fixerat på bokstaven *u* i ordet ”four”. Texten säger att endast fyra till fem bokstäver kan ses med 100 % noggrannhet (runt fixeringspunkten). Under den textraden visas en illustration på hur suddiga resterande bokstäver blir när ögonen fokuserar på bokstaven *u*. En färglagd synskärpsindikator visas som anger hur mycket som kan ses med en procentuells noggrannhet. Den slutsats som kan dras av Figur 3.6 är att relevant information ska finnas tätt intill fixeringspunkter för att på så sätt minimera sackadlängderna.

Om en internetanvändare vill se en TV-tablå på internet kan han eller hon exempelvis gå in på http://kanal5.se/web/guest/epg. I skrivande stund ser den TV-tablå som Kanal 5 tillhandahåller ut som i Figur 3.7. Skärmdumpen är tagen den 26 april 2011 kl. 14.36.

Figur 3.7 Skärmdump på TV-tablån från Kanal 5

Kanal 5s TV-tablå ger en bra översikt av programmen som visas på TV idag. Det som visas just nu är färgat blått, det som redan har passerat är färgat grått och det som visas senare i en mer beige färg. Vill användaren veta mer om ett program kan han eller hon klicka på programmets namn. I Figur 3.8 är *14:30 Vänner* klickat. Det dyker då upp en popup-ruta tätt intill där användaren har klickat som informerar användaren om genre på programmet, programmets övergripande handling, vad just detta avsnitt handlar om samt vilket säsongsnummer och avsnittsnummer avsnittet har. Med andra ord får användaren relevant information i närhet till klickningen och på så sätt har längden på sackaden som måste göras minimerats.

När en användare ska klicka på en länk eller dylikt, har han eller hon sin fixeringspunkt just där klickningen skedde. Då användaren klickar på *Vänner* har han eller hon fixerat blicken någonstans mitt på ordet *Vänner* och då popup-rutan kommer upp tätt intill fixeringspunkten ses denna enkelt. Hade det dykt upp en popup-ruta i övre vänstra bildkant hade användaren inte lika enkelt sett den, i värsta fall hade användaren klickat på länken ett flertal gånger och inte förstått att det dykt upp någon tilläggsinformation alls.



Figur 3.8 Skärmdump på TV-tablån från Kanal 5 där 14:30 Vänner är klickat

3.2.2 Erfarna och icke erfarna användare

När en hemsida eller applikation skapas är det viktigt att tänka på slutanvändarna, vilket presenteras i den andra regeln i de *gyllene reglerna* i avsnitt 3.1.1. Är hemsidan eller applikationen gjord för samtliga som äger en TV och vill veta vad som går på de allra vanligaste kanalerna, eller är den gjord för ett fåtal specifika användare? I de specifika fallen, som ett företags spårningsfunktion eller lagerstatus kan användargränssnittet vara mer komplicerat då det endast är ett fåtal användare av applikationen och de kommer efter en stunds integration med applikationen anses vara erfarna användare. De kommer alltså veta var popup-rutor och liknande information kommer att dyka upp på skärmen.

3.3 Färger

Färger kan vara attraktivt för användaren, de kan ofta bidra till en outtalad förståelse för programmet och bidra till undermedveten kunskap. Dock kan det även leda till missförstånd vid fel färgval [43]. Exempelvis kan det vara en bra idé att undvika färgerna grönt och rött på grund av färgblindhet. 8 % av männen och nästan 1 % av kvinnorna i västvärlden har någon form av permanent färgbrist i sin syn. Många andra kan ha temporärt nedsatt färgseende på grund av sjukdomar eller medicinering. Med det menas att de exempelvis kan ha svårt att skilja på rött och grönt [43].

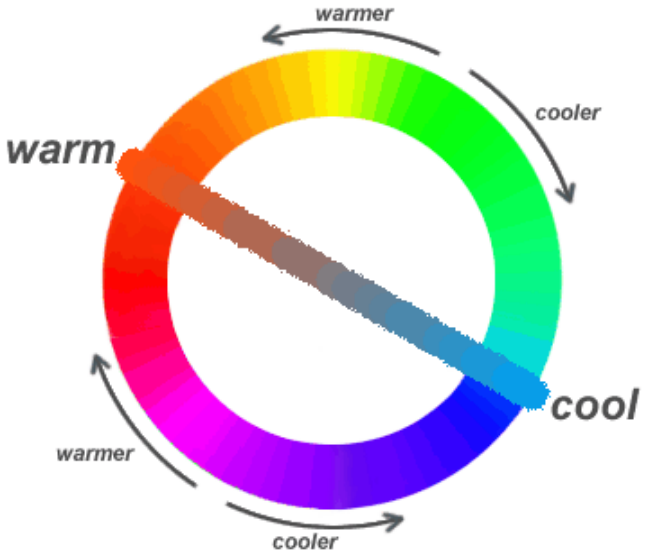
Färgerna rött och grönt kan kombineras då rött har ett syfte att avbryta och grönt att godkänna en händelse. Om syftet är att avbryta eller godkänna kan det då även vara bra att visa symboler som ett kryss och en bock för att ytterligare förklara knapparnas åtgärder. Dock spelar även kultur en viktig roll och det är därför viktigt att tänka på vem som kommer använda symbolerna. I USA och Sverige betyder en bock ett godkännande och ett kryss avböjande, medan i Storbritannien kan en bock och ett kryss visa acceptans [44].

Antalet färger bör även begränsas. Ett flertal designguider rekommenderar maximalt fyra färger i en bildruta och sju färger i en sekvens av rutor (displayfönster). Erfarna användare kan dra nytta av fler färger men för nybörjare kan det vara förvirrande [43]. När färger används i praktiskt syfte, exempelvis för att skilja på element bör maximalt en av färgerna användas [45]. Om båda används bör det finnas något annat kännetecken på elementen som underlättar när användaren behöver skilja gröna och röda element åt, till exempel, genom en ikon eller en beskrivande text.

Ljusa bakgrundsfärger, såsom vitt, beige och ljusgrått skiljer sig mycket från mörka bakgrundsfärger. Mörka bakgrundsfärger ger ofta en känsla av dysterhet eller kantighet, de används sällan i grafiska användargränssnitt. Textfält och knappar ser bättre ut på en ljus bakgrund och därför används ljusa bakgrundsfärger oftare i grafiska användargränssnitt. Om designern av det grafiska användargränssnittet ändå väljer att ha mörka bakgrunder är det viktigt att tänka på att ha ljusa förgrunder, och mörka förgrunder på ljus bakgrund [46].

3.3.1 Varma och kalla färger

Röd, orange, gul, brun och beige anses vara varma färger, medan blå, grön, lila, grå och vitt (i stora mängder) anses vara kalla färger. Hemsidor och gränssnitt som behöver ge en känsla av anständighet eller en försiktighetsfaktor använder ofta främst kalla färger, särskilt blått. En kombination av kalla och varma färger kan vara effektivt för att uppnå en balanserad känsla/intryck [46]. Figur 3.9 visar varma och kalla färger [47].



Figur 3.9 Varma och kalla färger

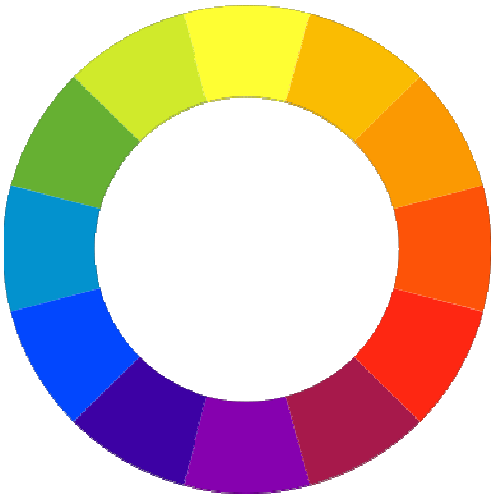
Varma färger väcker en känsla av energi och nyfikenhet. Den varmaste av de varma färgerna är rött och associeras ofta med energi, varning och aktsamhet. De allra viktigaste trafikskyltarna är röda och gula, exempelvis varningsmärken, väjningspliktsmärken och förbudsmärken. De mer underordnade trafikmärkena såsom påbudsmärken och parkeringsmärken är blå och vita [48].

Den kallaste av de kalla färgerna är blått och associeras ofta med lugn, pålitlighet, koncentration och andlighet. Majoriteten av de väletablerade hemsidorna och programmen använder kalla färger för sina användargränssnitt.

Det är dock inte bara färgen som påverkar oss[6], även nyanser och kontraster bidrar till känslotillstånd. Exempelvis kan en ”babyblå” färg ge oss lugn och harmoni, medan en ”skrikig” blå färg med hög kontrast kan ge motsatt effekt. Hög kontrast väcker spänning, styrka och djärvhet, medan låga kontraster är lugnande och avslappnande [46].

3.3.2 Komplementfärger

Valet av textfärg på en färgad bakgrund bör inte vara varandras komplementfärger då människors ögon har svårt att läsa komplementfärger [49]. Undvik till exempel ljusblå text på ljusröd eller orange bakgrund [46]. Figur 3.10 visar ett färghjul där färger som är på varandras motsatta sida är varandras komplementfärger [52].



Figur 3.10 Färghjul. En färgs komplementfärg återfinns på motsatt sida av cirkeln

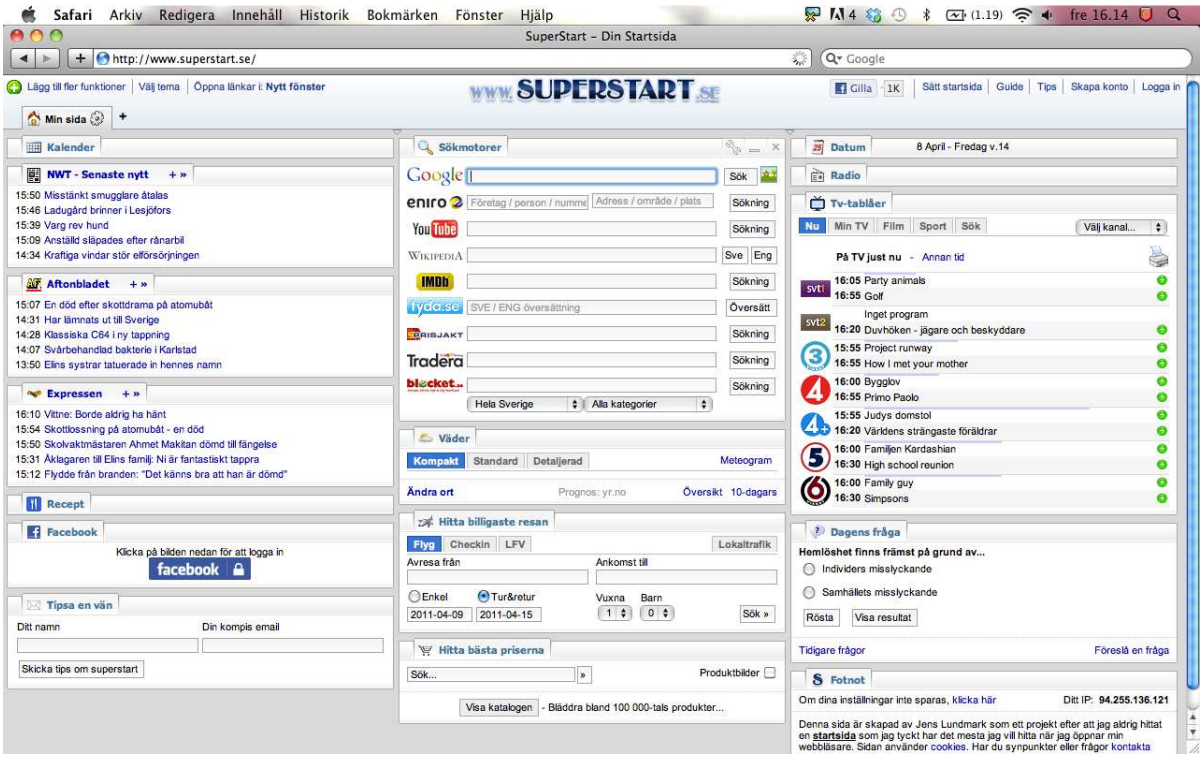
[6] Läs mer om färger och dess betydelse på http://www.color-wheel-pro.com/color-meaning.html

3.3.3 Summering

Människor påverkas mycket av färger. Det finns åtskilliga böcker och studier i ämnet. Dock är det sällan slutanvändare reflekterar över färgerna i en applikation, ”det bara känns bra”. Det är därför vår uppgift, som gränssnittsdesigners att använda oss av rätt färger för att göra vår applikation så attraktiv som möjligt. En bra hemsida eller applikation kan enkelt fördärvas av fel färger, eller för många färger. Åter igen är det viktigt att tänka på slutanvändarna och på budskapet. Är applikationen gjord för att arbeta i längre stunder eller för att fånga uppmärksamhet och nyfikenhet? Ska hemsidan vara anständig och neutral eller ska den vara framfusig och väcka känslor?

3.4 Informationsbelastning

När det talas om informationsbelastning kan det menas den totala informationsbelastning människor utsätts för från olika källor, till exempel genom mobiler, mejl, internet och intranät. Det kan även talas om informationsbelastningen från en viss applikation eller hemsida. Med applikation menas informationen från applikationens användargränssnitt, hur mycket information användaren utsätts för.



Figur 3.11 Skärmdump på Superstart.se

Ovan visas en skärmdump på sidan http://www.superstart.se/ från den 8:e april 2011. Vid första anblick kan användaren få en chock då det kan upplevas som för mycket information på en gång. Om det är informationsöverbelastning eller inte beror på hur mycket användaren har besökt sidan och varför sidan besöktes. Desto erfarnare användare och desto oftare de använder sidorna/applikationerna desto mer information kan de belastas med. Efter att ha klickat runt lite på sidan inses det att det är möjligt att filtrera bort oönskad information och anpassa den efter sina egna behov.

Färgerna är lugna och kalla och när användaren hovrar över olika länkar får han eller hon upp mer information om dessa, vilket är en bra funktion. Trots det är det för mycket information på en och samma gång.

Användaren av denna sida ges möjlighet att kasta om blocken, ta bort och lägga till block så som önskas samt att skapa ett konto. Syftet med att skapa ett konto är att vid ett senare tillfälle eller från en annan dator kunna logga in och få se sin egen anpassade sida. Då någon timme spenderats på sidan och format den efter det egna behovet/intressen så blir den användbar.

Dock är det ingen sida som bör besökas om användaren snabbt vill ta reda på specifik information, till exempel veckans veckonummer.

3.5 Summering

Kapitel 3 har berört ett antal designaspekter. Det har diskuterats kring grafiska användargränssnitt, vad det är och vad som kännetecknar ett bra sådant. Kapitlet har berört ögonrörelser, hur våra ögon rör sig när de läser eller betraktar en bild vilket i många fall kan jämföras med att använda en applikation. Det har även diskuterats vilka färger som bör användas och vilka som bör undvikas när hemsidor eller applikationer skapas. Dessutom har kapitlet kort diskuterat informationsbelastning.

De *gyllene reglerna*, som nämndes i avsnitt 3.1.1, hur våra ögon rör sig samt färgläran har varit en utgångspunkt när andras hemsidor har studerats. De finns lika eller snarlika riktlinjer i ett flertal grafiska användargränssnittguider och även om reglerna kan verka vara sunt förnuft, är det bra att studera dem då det är lätt att glömma vissa aspekter när grafiska användargränssnitt utvecklas.

4 Prototyper av det grafiska användargränssnittet

Detta kapitel kommer att beröra själva implementationen av programmet. Olika alternativ för utseende av spårbarhetsgrafen kommer att presenteras där det diskuteras för- och nackdelar med respektive alternativ. Grafens olika komponenter kommer att diskuteras samt hur dess information kan presenteras.

4.1 Spårbarhetsgrafens komponenter

Detta avsnitt behandlar spårbarhetsgrafens två kärnkomponenter; noden och pilen. Båda är viktiga informationsbärare i grafen och representerar ämnena samt operationerna i tillverkningsprocessen. Noderna representerar ämnen och pilarna representerar operationer. För att få ett avstånd mellan samtliga komponenter som känns luftigt men som ändå är platsbesparande krävs en formel som tar hänsyn till komponenternas bredd och höjd. Detta då noderna kan växa och krympa beroende på vilken storlek användaren har valt. Detta avsnitt behandlar därför även hur utplacering av komponenterna som fungerar bra oberoende av storleken på noderna kan göras.

4.1.1 Noden

Noden är informationsbäraren om en råvara, smälta eller ett göt i spårbarhetsgrafen. Den innehåller relevant information om ämnet som exempelvis dess namn och vikt. Dess namn visas direkt i noden (se Figur 4.1 där namnet är *Smälta 2*) medan information såsom vikt och artikelnummer kan erhållas genom att antingen klicka på eller föra muspekaren över noden. Hur denna information skall presenteras diskuteras i avsnitt 4.4. Nodens färg beror på vilken typ av ämne den representerar, exempelvis råvaror, smältor och göt. De färger som används i detta kapitel är valda för att ge en hög läsvänlighet. Med det menas att slutanvändaren själv kan bestämma vilka färger han eller hon vill använda när de startar applikationen som ritat upp en graf. Syftet med att färglägga noderna är att, som det diskuterades i Kapitel 3, ge erfarna användare ett ytterligare sätt att arbeta på. Mer erfarna användare kan urskilja vilket

typ av ämne noden representerar utan att läsa texten i den, vilket leder till att de får en snabbare överblick.

Figur 4.1 Nod med texten Smälta 2

Om muspekaren förs över eller klickar på noden så växer dess höjd och bredd med en bestämd storlek. Det förinställda värdet som valts är att den växer med 15 % i båda riktningar. Detta används för att tydliggöra vilken nod som är markerad. Värdet valdes då det var en lagom förstoring för att användaren ska kunna se skillnad på en vald och icke vald nod. Se Figur 4.2 för en illustration av skillnad i storlek mellan en markerad och inte markerad nod. Texten i noden växer även den proportionerligt med noden.

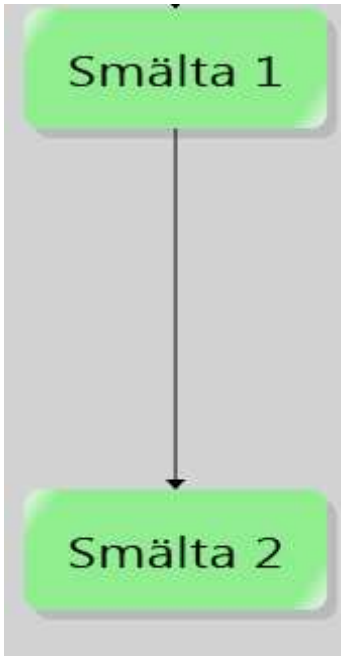
Figur 4.2 Den vänstra noden visar dess storlek när den blivit klickad på eller muspekaren hovrar över den. Noden till höger visar dess storlek i övriga fall

För att göra noderna lite mer attraktiva att titta på så används skuggningar och belysningar. Skuggan faller som om noderna vore belysta från det övre vänstra hörnet. Belysningarna ligger på nodens övre vänstra hörn och nedre högra hörn. Belysningarna kan även kallas för blänkeffekt. Dessa effekter ger ett djup i bilden och känns därför lite modernare. Det kan vara lite svårt att få en överblick av dessa effekter i Figur 4.2. En bättre överblick visas i Figur 4.7.

4.1.2 Pilen

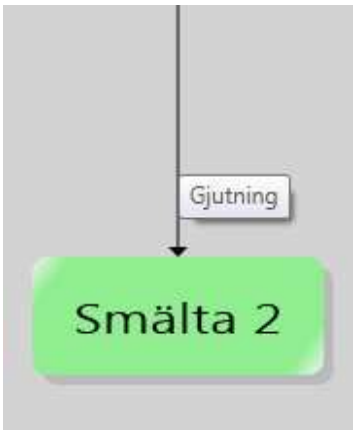
Pilen mellan noderna visare beroendet mellan noderna. Den nod som pilspetsen pekar på har sitt ursprung i den nod som pilen har sin början i. Se Figur 4.3 för ett exempel. En nod kan ha

flera pilar som pekar på den vilket betyder att den har sitt ursprung i flera ämnen. På samma sätt kan en nod peka till flera andra noder.



Figur 4.3 Pil mellan noderna Smälta 1 och Smälta 2

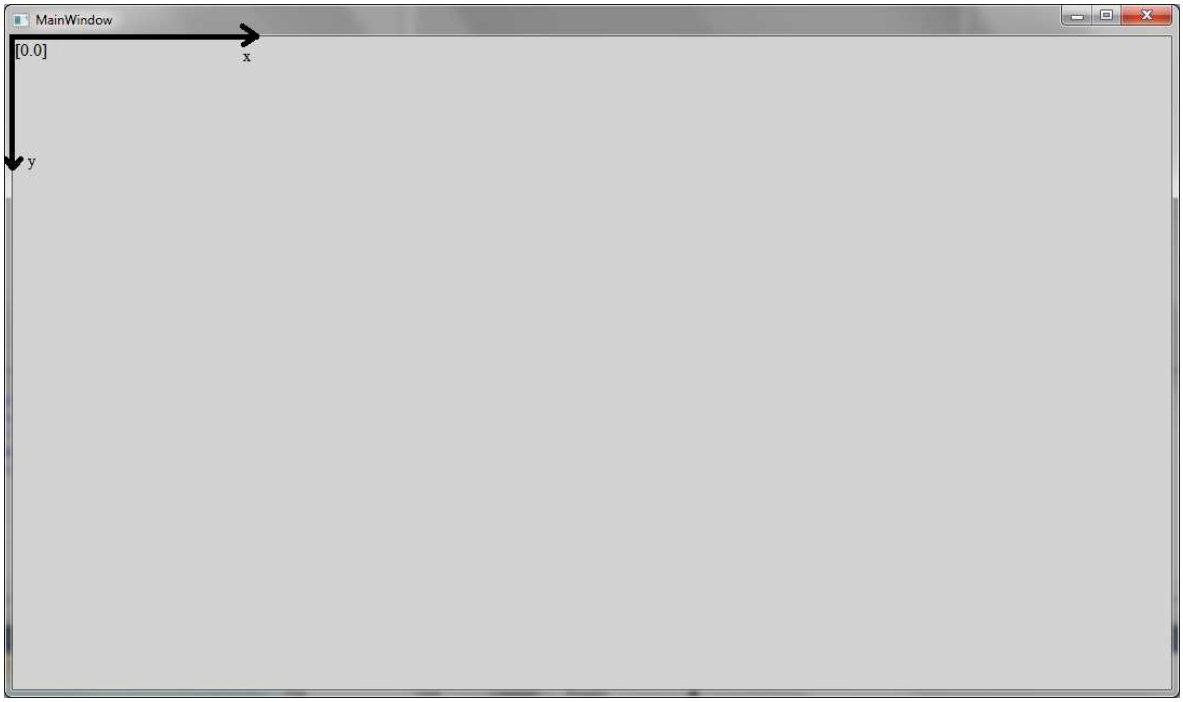
Om användaren för muspekaren över eller klickar på pilen så dyker ytterligare information upp om operationen som pilen ska representera (exempelvis smältning eller gjutning). I Figur 4.4 hålls muspekaren över pilen till *Smälta 2*. Informerande text om operationen, i det här fallet *Gjutning*, dyker då upp. Denna information skulle även kunna presenteras i de statiska informationsrutorna som diskuteras i avsnitt 4.4.2, dock anses det onödigt då användarna efter en kort stunds interaktion med programmet förstår vad noderna står för och på så sätt deras operation. På så sätt minimeras onödig eller underförstådd information i den statiska informationsrutan.



Figur 4.4 Pil med informerande text

4.1.3 Positionering av komponenterna

I WPF så kan komponenter placeras ut i fönstret genom att ange x- och y-koordinater för komponenten. Hur detta koordinatsystem är uppbyggt visas i Figur 4.5. x-led växer åt höger och y-led växer nedåt. Detta måste tas i beaktande vid den automatiska utplaceringen av komponenterna.



Figur 4.5 Koordinatsystem för applikationsfönstret

Vid utplacering av noderna så bör hänsyn tas till nodernas bredd och höjd istället för att använda ett fixt värde som avstånd. Nedan presenteras de formler som har använts. Dessa ger ett lagom långt avstånd mellan noderna utan att få känslan av att grafen är alltför utdragen.

Koordinaterna för nodernas övre vänstra hörn fås av.

$$y = (nivå - 1) * nodens\ höjd * 1.4$$

$$x = nodens\ bredd * (kolumn * antal\ kolumner\ i\ längsta\ raden * 1.4 - 1)$$

Efter testning, valdes konstanten 1.4 för att få ett sådant avstånd mellan noderna att de inte skär varandra men att det inte heller blir ett för utdraget avstånd vilket innebär platsbesparande. Nivåerna representerar vilken rad noderna ska placeras på med startindex 1.

Nodernas skuggor består av en lika stor rektangel som noden själv som är gråfärgad och placeras ut enligt formlerna nedan.

$$y = nod.Y + 5$$

$$x = nod.X + 5$$

Koordinaterna för linjerna fås av följande formler där A är den nod som pilen utgår ifrån och B är den nod som pilen pekar på.

$$Yfrån = A.Y + A.Höjd$$

$$Xfrån = A.X + \frac{A.Bredd}{2}$$

$$Ytill = B.Y$$

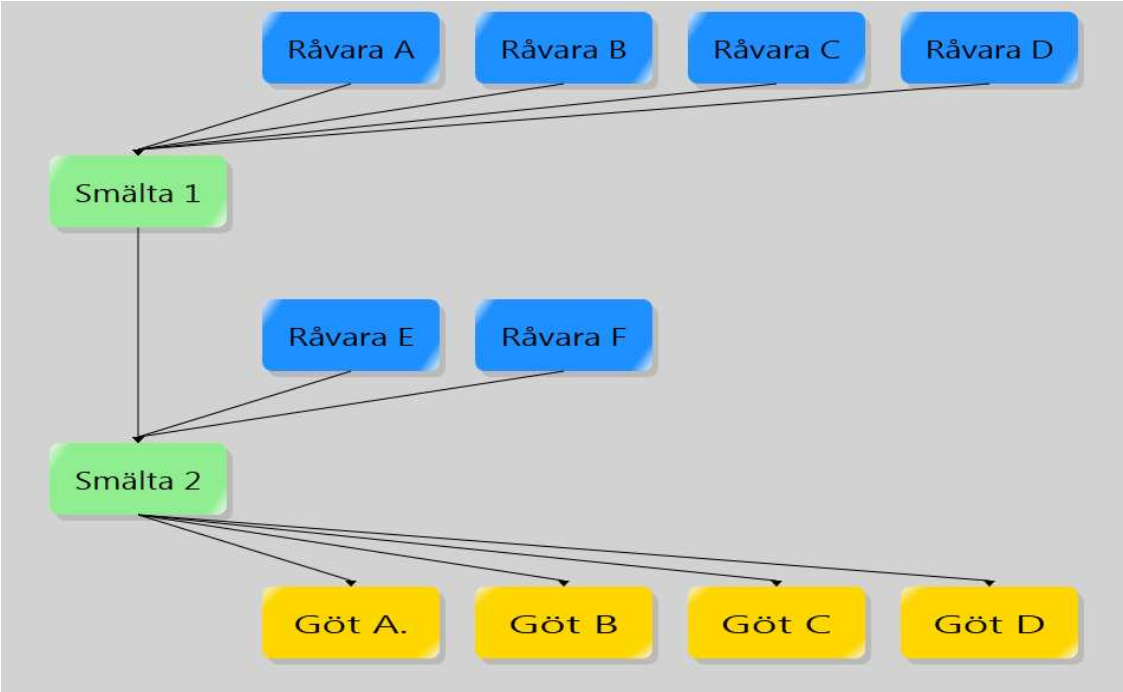
$$Xtill = B.X + \frac{B}{2}$$

4.2 Utseende för spårbarhetsgrafen

Detta avsnitt tar upp tre olika alternativ på hur grafen kan presenteras. Det första är en vertikalt icke-centrerad version där smältorna lokaliseras till vänster och de övriga ämnen till höger. Alternativ två innebär en vertikalt centrerad version där en symmetri eftersöks för att få en jämn fördelning av grafen. Alternativ tre innebär en horisontell variant av grafen och presenteras som en centrerad variant. Den skulle dessutom vara möjlig att presentera som en icke-centrerad variant. Dessa tre varianter har valts för att begränsa antalet prototyper. Om den vertikala eller horisontella skulle kombineras med den centrerade eller icke-centrerade grafen så skulle sex olika prototyper fås. Så många eller ännu fler prototyper kan vara bra i ett tidigt stadie, men vid presentation av dessa för kunden är det bra att begränsa antalet val. Annars kan kunden bli förvirrad och får svårt att välja en prototyp för fortsatt arbete.

4.2.1 Vertikalt icke-centrerad version

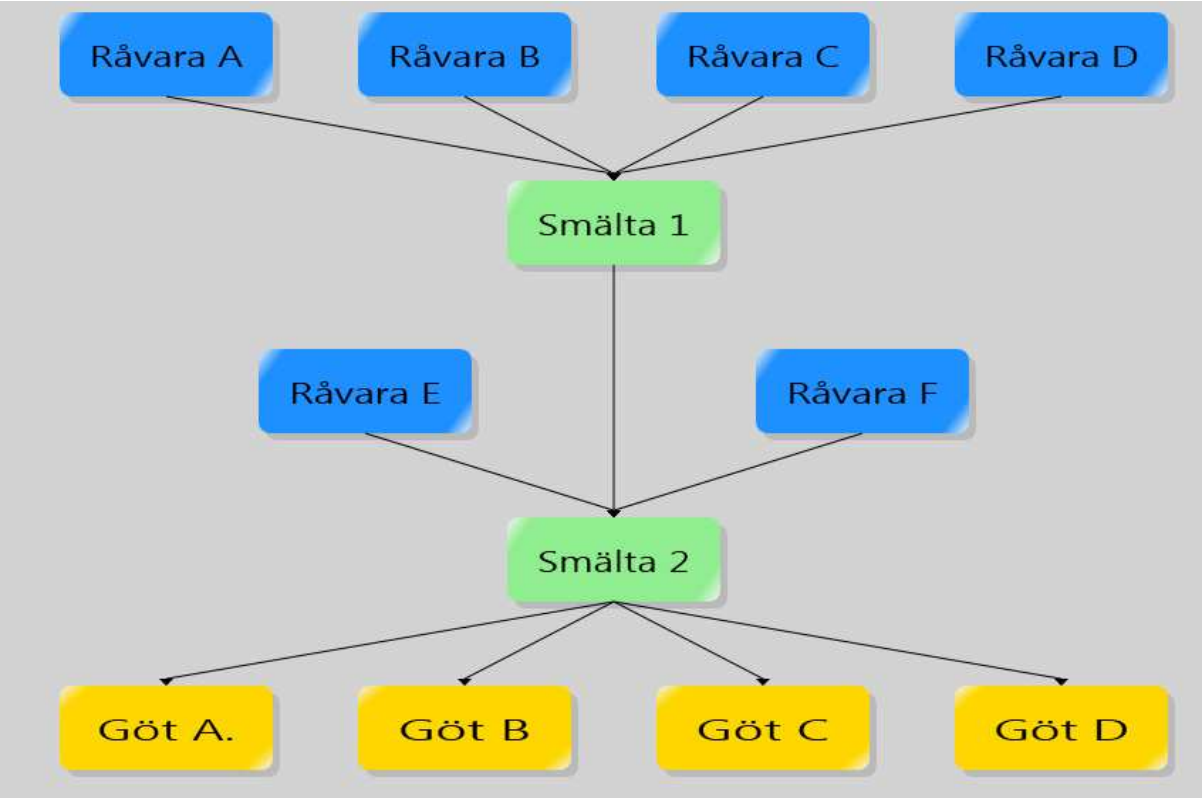
Den första versionen som kan ses i Figur 4.6 är den icke-centrerade versionen. Smältor är lokaliserade till vänster i grafen och råvaror samt göt placeras till höger. På så vis får användaren god översikt av alla smältor. Nackdelen är att grafen blir utspridd och detta leder till längre ögonrörelser än vid jämförelse med en centrerad version. Långa ögonrörelser diskuterades i Kapitel 3 och det är något som i största möjliga utsträckning bör undvikas.



Figur 4.6 Vertikalt icke-centrerad version

4.2.2 Vertikalt centrerad version

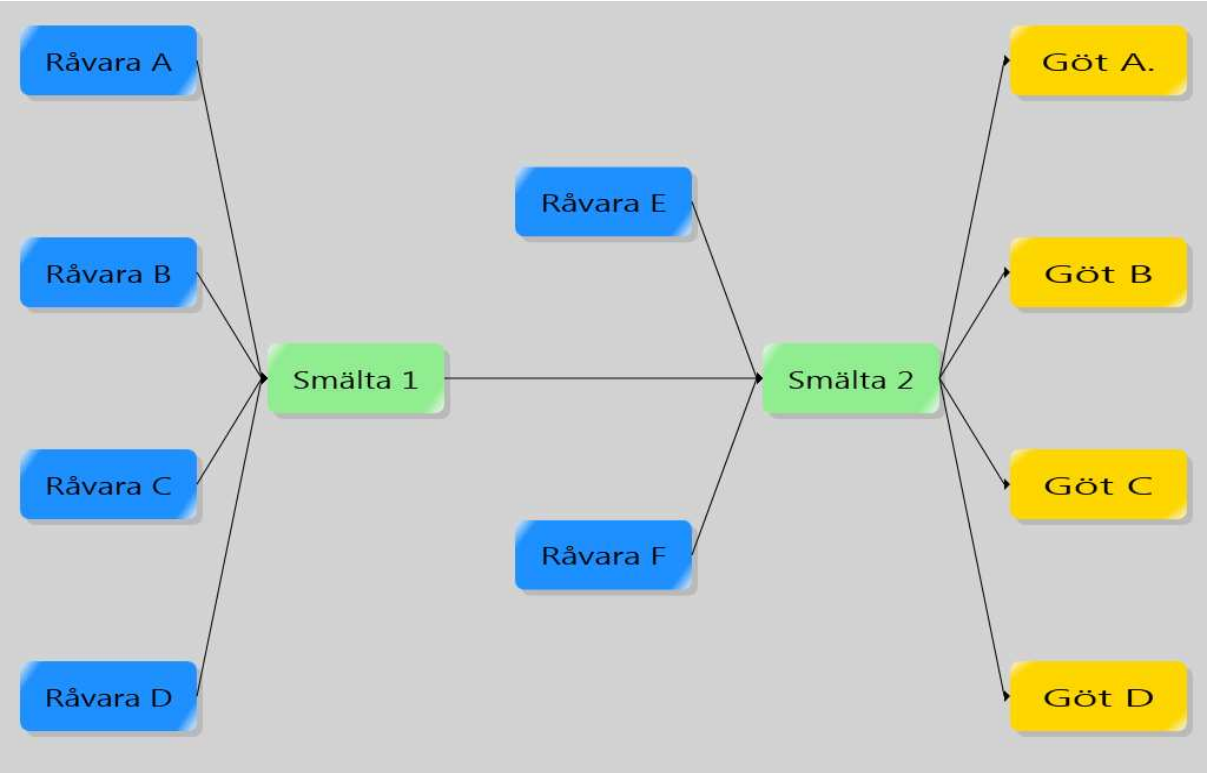
Den andra versionen, som visas i Figur 4.7, centrerar noderna i grafen så mycket som möjligt. Syftet med att centrera grafen är att minska ögonrörelser i största möjliga utsträckning för att öka komforten. När användaren fokuserar på *Råvara D* i Figur 4.6 så märker denne snart att det leder till förflyttning av blicken över hela skärmen för att följa pilen som leder till *Smälta 1*. Så långa ögonrörelser är inte lika vanliga i Figur 4.7.



Figur 4.7 Vertikalt centrerad version

4.2.3 Horisontellt centrerad version

Den tredje varianten som kan ses i Figur 4.8 är en horisontellt växande graf. Den kan användas både som en centrerad och icke-centrerad graf. I Figur 4.8 visas den centrerade varianten. Den horisontella utläsningen av grafen kan kännas naturlig till en början, då västvärlden i regel läster från vänster till höger, dock använder den inte musens rullhjulsfunktion särskilt väl då den växer i sidled och inte i vertikalt led. Som det diskuterades i Kapitel 3 gör det ingen nämnvärd skillnad i vilken riktning som användaren läser eller betraktar en bild.



Figur 4.8 Horisontellt centrerad version

4.2.4 Summering

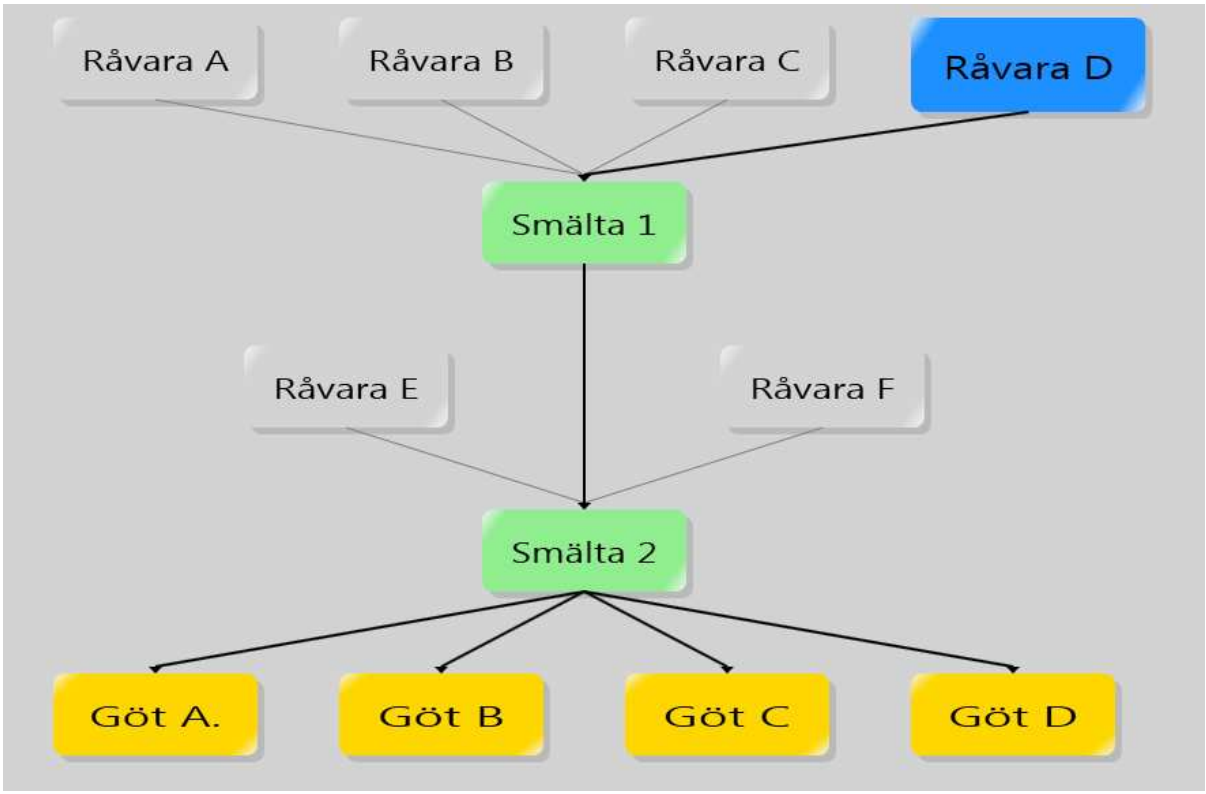
Detta avsnitt har behandlat utseendet på grafen. I det här fallet hur noderna ska placeras, centrerat eller ej centrerat. Det har även diskuterats olika lösningar för hur grafen kan växa, horisontellt eller vertikalt. Sogeti har valt att använda den vertikalt centrerade grafen så i fortsättningen kommer det att utgås från den när andra designaspekter diskuteras.

4.3 Spårning i grafen

När användaren för muspekaren över eller klickar på en nod i grafen ska han eller hon få information om nodens ursprung och dess framtida riktning. Syftet med detta är att snabbt ge användaren en grafisk översikt av en nod. Detta avsnitt behandlar hur programmet ska presentera denna spårning. Två prototyper visas där den ena färgar de icke berörda noderna gråa och den andra gör dessa transparenta.

4.3.1 Icke berörda noder färgas grå

I Figur 4.9 har användaren klickat på *Råvara D*. Noden är 15 % större än de andra noderna för att göra det lite tydligare vilken nod som har klickats. De noder som är rakt nedstigande (i det här fallet *Smälta 1, Smälta 2, Göt A, Göt B, Göt C och Göt D*) eller rakt uppstigande (i det här fallet inga noder) behåller sin färg medan övriga noder färgas grå. Fördelen med denna lösning jämfört med den som presenteras i 4.3.2 är att läsbarheten inte försämras då texten behåller samma färg. Eftersom bakgrunden också är grå så upplevs noderna ha smält in i bakgrunden så att användaren lättare kan fokusera på de väsentliga noderna. Nackdelen är den snabba överblicken går förlorad då gråfärgade noderna inte kan identifieras med hjälp av dess färg.



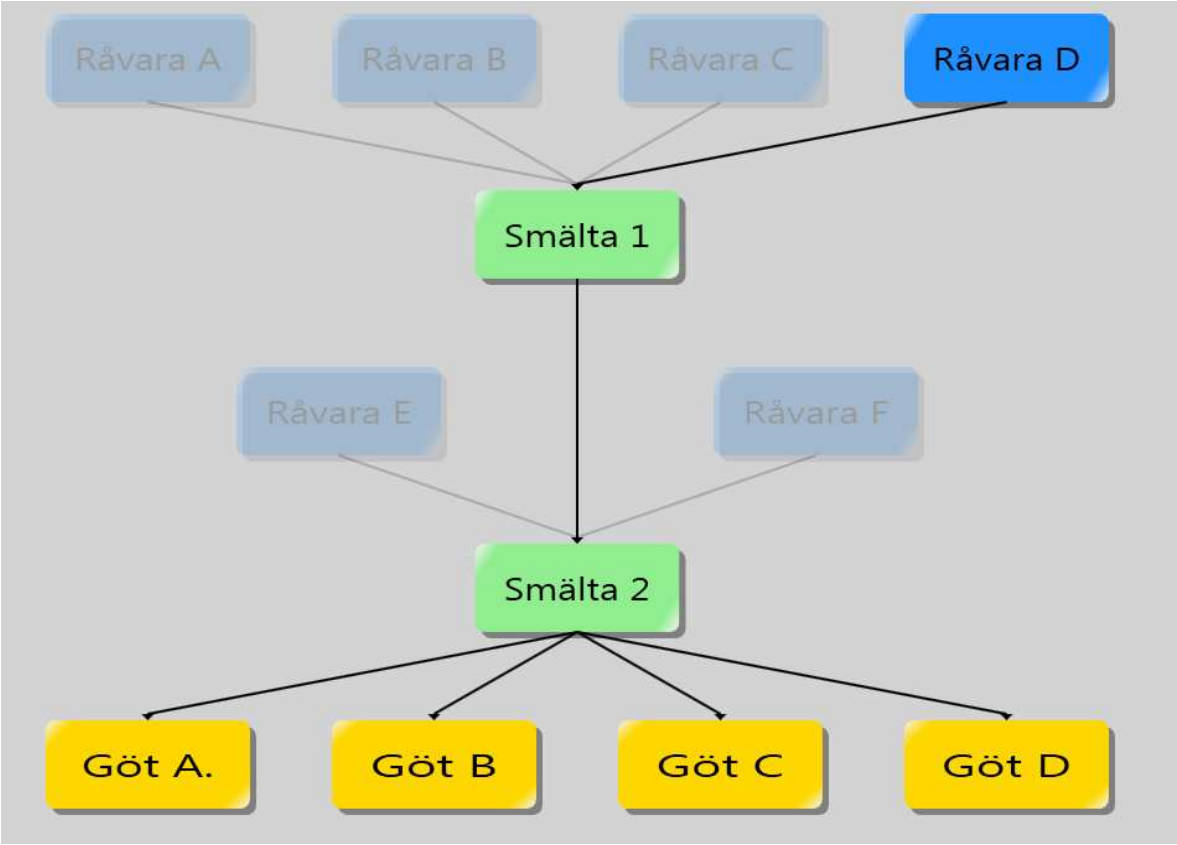
Figur 4.9 Icke berörda noder färgas grå

De gråfärgade noderna behåller sin skuggning och belysning och därför ser användaren ändå konturerna på noden. Genom att titta noggrant mitt på en grå nods övre kant ses ingen linje

eller skuggning, den delen av noden har exakt samma färg som bakgrunden. Anledningen till att användaren ändå uppfattar konturerna av en rektangel är för att hjärnan försöker tolka synfältet och hitta en logisk förklaring på vad det föreställer. Människor har ett medfött behov av att få struktur på omvärlden och därav kan hjärnan hjälpa oss att se linjer eller mönster som egentligen inte finns [50].

4.3.2 Icke berörda noder blir transparenta

I Figur 4.10 visas utseendet på grafen när användaren klickat på noden *Råvara D*. De noder som är rakt nedstigande och rakt uppstigande behåller sin färg, medan övriga noder blir transparenta med en opacitet på 20 %. Detta tydliggör vilka noder som inte är beroende av den valda noden. Värdet 20 % valdes då det fortfarande går att urskilja nodens färg men den tar mindre uppmärksamhet i anspråk. Nackdelen är att det blir något svårare att se texten i noderna som är transparenta. En fördel är att trots att resterande noder av typen *Råvara* är transparenta, kan användaren ändå identifiera nodtypen med hjälp av dess färg. Jämför gärna med varianten som presenteras i 4.3.1. En annan fördel med de transparenta noderna är att de känns mer nytänkande och de ser attraktivare ut.



Figur 4.10 Icke berörda noder blir transparenta

4.3.3 Summering

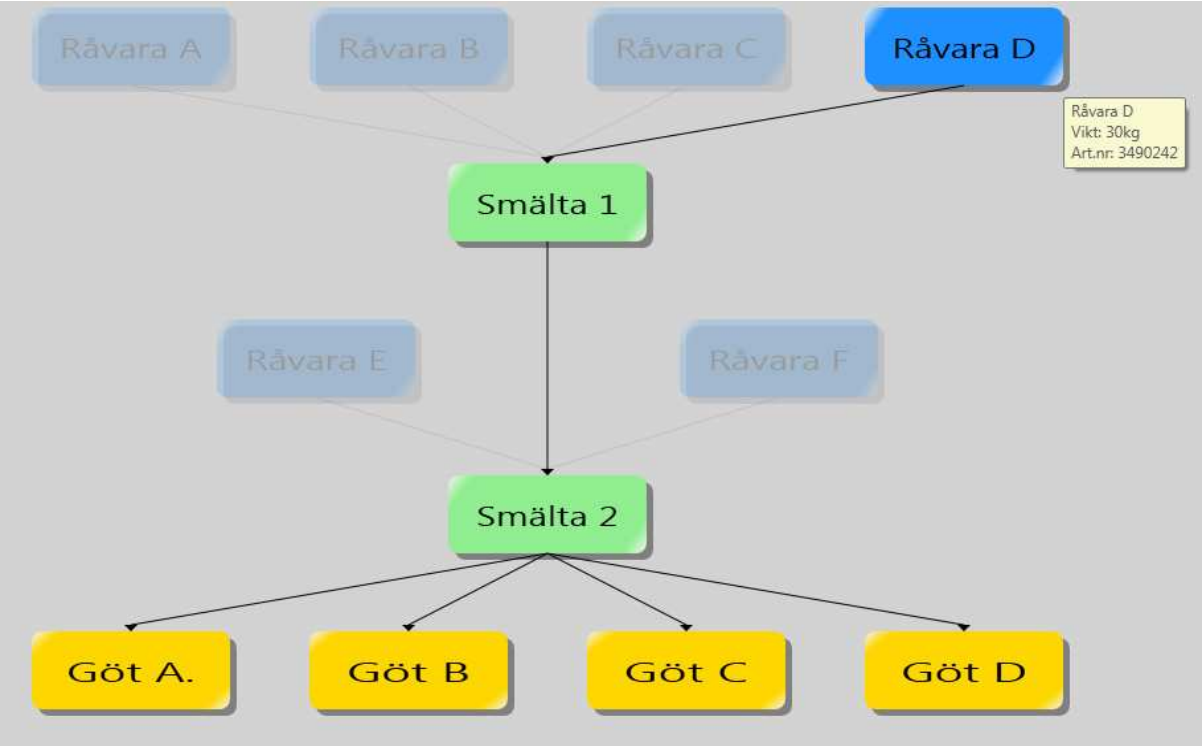
Att grafiskt kunna spåra grafen kan vara praktiskt för att snabbt få en överblick över var ett visst ämne kan sägas vara inblandat i tillverkningsprocessen. I detta avsnitt har det diskuterats hur denna överblick kan fås på bästa sätt och två olika prototyper har presenterats. Den ena färgar de icke berörda noderna gråa och den andra gör dem transparenta. Då Sogeti i nuläget anser att de transparenta noderna ser mer attraktiva ut än de gråfärgade valdes det att fortsätta arbeta på den lösningen.

4.4 Presentation av information

Detta avsnitt tar upp hur information som inte presenteras i nodernas textruta ska presenteras. Information som inte visas i nodens textruta kan vara vikt, artikelnummer samt annan viktig information för slutanvändaren. Tre prototyper på hur denna information kan presenteras. Det första alternativet är en dynamisk informationsruta som ändrar position beroende på vilken nod som väljs. De andra två alternativen är informationsrutor med en fast position. En informationsruta presenteras till höger om grafen och den andra under grafen. Dessa två rutor visas oavsett om informationen eftersöks eller inte.

4.4.1 Dynamiskt placerad informationsruta

Det första alternativet är en informationsruta med en dynamisk position som varierar beroende av vilken nod som eftersöks om information. Informationen visas, i tät anslutning till noden, när användaren antingen hovrar över eller klickar på noden. Om användaren hovrar över noden så ska informationen försvinna när muspekaren lämnar noden. Klickar däremot användaren på noden så skall information dröja kvar till det att höger musknapp klickas ned på bakgrunden eller att en annan nod klickas på. Figur 4.11 visar en sådan informationsruta för noden *Råvara D* när användaren har fört muspekaren över denna.



Figur 4.11 Dynamiskt placerad informationsruta för noden Råvara D

En fördel med denna lösning är att fönstret inte behöver anpassas efter en informationsruta som ständigt ska visas. Det gör att mer utrymme ges till noderna och leder till en bättre överblick. En annan fördel är att informationsrutan dyker upp i tät anslutning till noden, det vill säga användarens senaste fixering och därför behöver användaren göra en mindre sackad än vad han eller hon kommer att behöva göra till de statiska informationsrutorna som presenteras i avsnitt 4.4.2. Detta kan jämföras med TV-tablån från Kanal 5 som finns i Kapitel 3, där en informationsruta med liknande funktionalitet presenteras.

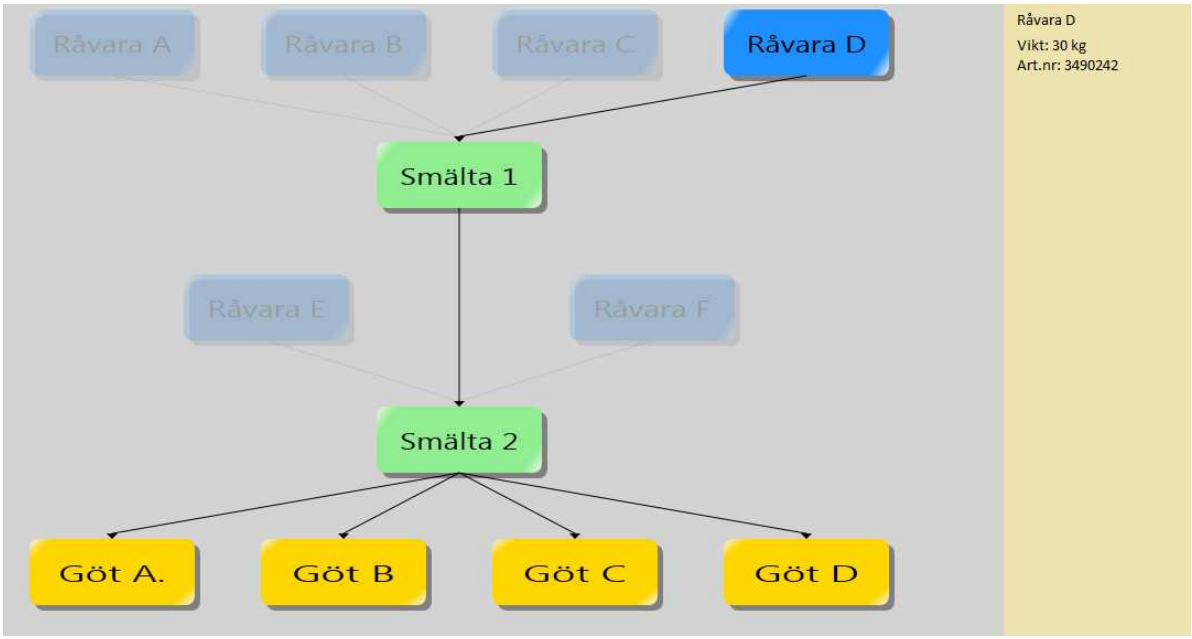
Om informationsrutan innehåller en stor mängd information kan det vara en mindre bra idé att ha rutan dynamiskt. Detta på grund av att rutan växer i takt med innehållet och om rutan täcker en stor del av grafen kan det anses besvärande.

4.4.2 Statiskt placerad informationsruta

En annan variant för att presentera information är att använda en informationsruta som ständigt visas även om en nod är markerad eller inte. Denna är något enklare att implementera

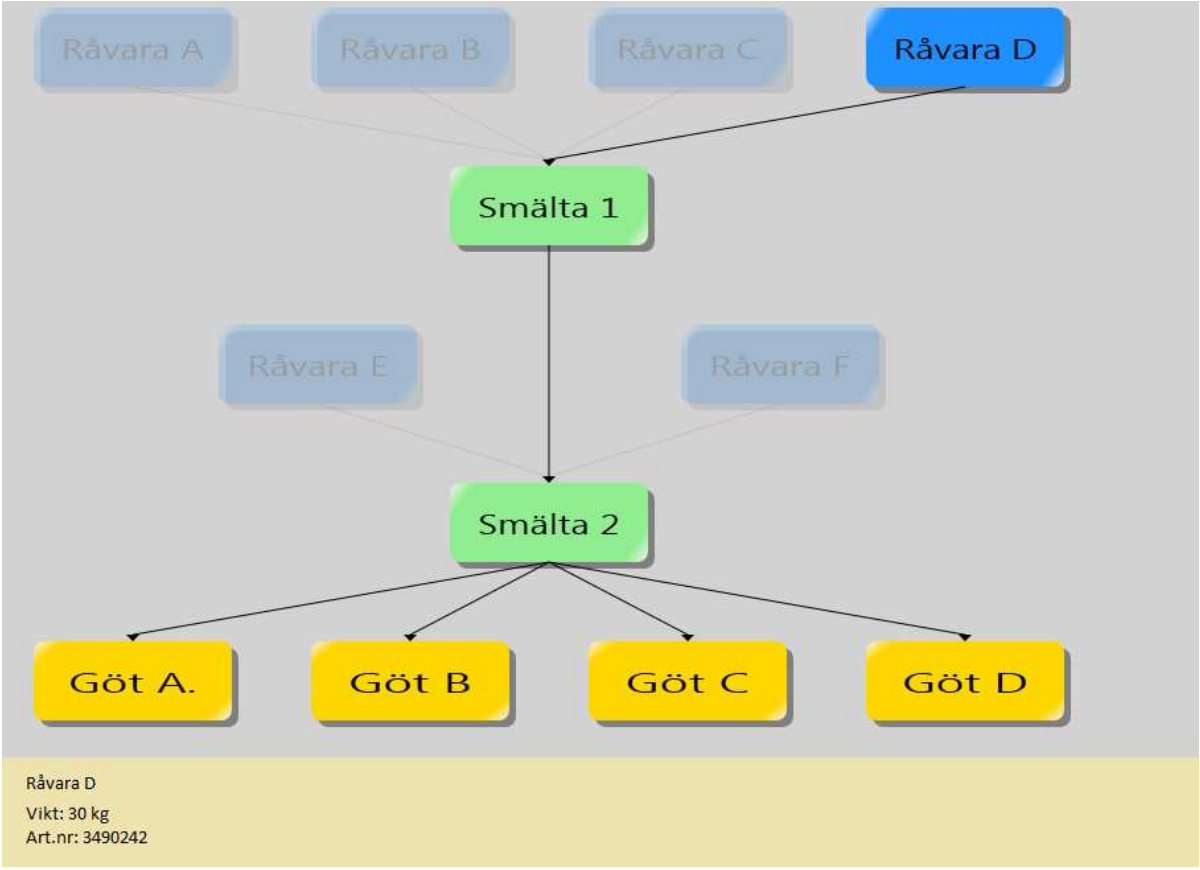
då implementeraren vid start av programmet anger dess position och behöver därefter inte ändra denna. En nackdel är att mindre utrymme ges för noderna och överblicken minskar något. Fördelen med denna lösning är att användaren alltid kan vänta sig att all information presenteras på en enda plats.

I Figur 4.12 visas en variant där rutan placeras till höger i fönstret. Noden *Råvara D* är klickad och dess information presenteras i informationsrutan. Ytterligare en nackdel är att noderna kan hamna långt ifrån informationsrutan vilket leder till en lång ögonförflyttning. I Figur 4.12 behöver användaren flytta blicken över hela skärmen för att få information om *Göt A*.



Figur 4.12 Statisk placerad informationsruta placerad till höger i fönstret

I Figur 4.13 visas en variant där rutan placeras i botten av fönstret. Noden *Råvara D* är även här markerad och dess information presenteras i informationsrutan. En nackdel är att den tar upp mycket utrymme på skärmen på grafens bekostnad då dess utrymme minskas. En annan nackdel är att såsom i Figur 4.12 blir långa ögonförflyttningar beroende på vilken nod som har klickats. I detta fall då *Råvara D* är klickat, måste användaren återigen förflytta ögonen från en ände av skärmen till en annan. Ytterligare en nackdel är att det leder till många scrollningar i informationsrutan om informationen består av många radbrytningar.



Figur 4.13 Statiskt placerad informationsruta placerad nederst i fönstret

4.5 Summering

I detta kapitel har implementationen av spårbarhetsgrafen diskuterats. Spårbarhetsgrafen innehåller i huvudsak två olika komponenter, noder och pilar. Noder är informationsbäraren om ämnena som behandlas i tillverkningsprocessen. Pilarna i sin tur innehåller information om operationerna som utförs.

Olika prototyper på hur spårbarhetsgrafen valts att presenteras och dess komponenter har visats. Det har även diskuterats samtliga prototypers för- och nackdelar med avseende på användarvänligheten och lärdomar som getts av Kapitel 3.

De färger som användes för noderna i detta kapitel är inte slutgiltiga utan slutanvändarna kommer själva att få bestämma vilka färger de vill ha på de olika typerna av ämnen. Detta då

de kanske redan har ett system för färgmärkning. De kanske är vana att se råvaror som gröna och då kan de välja att fortsätta göra på det viset.

I samråd med Sogeti valdes den dynamiska informationsrutan. Den dynamiska rutan kommer att innehålla kortare information. Den fullständiga informationen om en nod skickas ut från användarkontrollen när användaren klickar på en nod. På så vis kan användaren själv bestämma hur informationen ska presenteras. Vad som klassificeras som kortare respektive fullständig information avgör slutanvändarna.

5 Gränssnitt mot XML-fil

För att kunna använda programmet som efterfrågats, det vill säga att användaren genom extern data ska kunna skapa en graf med vår användarkontroll, så måste det skapas en funktion för att läsa från denna källa. Tillsammans med Sogeti valdes det att det ska användas filer av typen XML för att läsa och skriva data. Detta kapitel presenterar hur detta görs.

5.1 Användning av XML-filer

I Figur 5.1 visas ett exempel på hur en XML-fil som skickas in i grafritaren kan se ut.

```
<?xml version="1.0" encoding="UTF-8" ?>
<graph>
  <typecollection>
    <!--Definiera typerna-->
    <type>
      <name>Råvara</name>
      <color>LightBlue</color>
    </type>
  </typecollection>
  <nodecollection>
    <!--Lägg till noder-->
    <node>
      <id>1</id>
      <text>Råvara A</text>
      <level>1</level>
      <type>Råvara</type>
      <information>10 kg</information>
    </node>
  </nodecollection>
</graph>
```

Figur 5.1 Exempel på utseende för XML-fil

Typer definieras mellan taggen <typecollection> och inom <type> där <name> är namnet på råvaran och <color> är färgen på noderna som är av den givna typen. Färgen kan antingen

vara de förbestämda färgerna från klassen Color i Systems.Windows.Media [51] eller från hexadecimala värden[7].

Noder definieras mellan taggen <nodecollection> och inom <node>. <id> är ett för varje nod unikt heltal som fungerar som en identifierare för noden. <text> är texten som ska stå synlig i noden, <level> är vilken rad som noden ska placeras och måste vara minst 1. <type> anger av vilken typ noden är och här ska namnet på typen stå. <information> är all övrig information om noden som ska visas i informationsrutorna. XML-filen resulterar i grafen i Figur 5.2 som består av en nod med texten *Råvara K* av typen *Råvara* som har färgen *LightBlue* från klassen Color.

Figur 5.2 Resultat av XML-fil

Om pilar mellan noder önskas lägga till så läggs dessa in mellan taggen <connectorcollection>. Ett exempel visas i Figur 5.3.

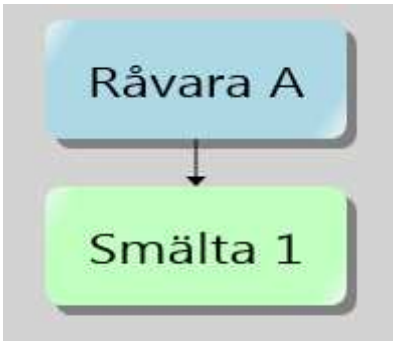
[7] Generera hexadecimala värden kan göras till exempel på http://www.colorschemer.com/online.html

```
<?xml version="1.0" encoding="UTF-8" ?>
<graph>
  <typecollection>
    <!--Definiera typerna-->
    <type>
      <name>Råvara</name>
      <color>LightBlue</color>
    </type>
    <type>
      <name>Smälta</name>
      <color>#C1FFC1</color>
    </type>
  </typecollection>
  <nodecollection>
    <!--Lägg till noder-->
    <node>
      <id>1</id>
      <text>Råvara A</text>
      <level>1</level>
      <type>Råvara</type>
      <information>10 kg</information>
    </node>
    <node>
      <id>2</id>
      <text>Smälta 1</text>
      <level>2</level>
      <type>Smälta</type>
      <information>9 kg</information>
    </node>
  </nodecollection>
  <connectorcollection>
    <!--Lägg till pilar-->
    <connector>
      <name>Smältning</name>
      <from>1</from>
      <to>2</to>
    </connector>
  </connectorcollection>
</graph>
```

Figur 5.3 XML-fil med både noder och pilar

Där har det lagts till en ny typ med namnet *Smälta* och getts en färg från ett hexadecimalt färgvärde. Som det har nämnts innan så kan både namnen i klassen Color och de hexadecimala värdena för färgerna kan användas. Ytterligare en nod med texten *Smälta 1* läggs till och placeras på rad 2. Därefter läggs en pil i <connectorcollection> inom <connector>. <from> anger vilken nod som pilen utgår från, där ska nodens id föras in. <to> anger vilken nod pilen ska peka på, även här ska nodens id anges. <name> är den text som

syns när användaren håller muspekaren över pilen. Resultatet av XML-filen ovan visas i Figur 5.4.



Figur 5.4 Graf med två noder och en pil

Det är viktigt att typerna och komponenterna definieras i följande ordning då noder beror av typ och pilarna beror av noderna.

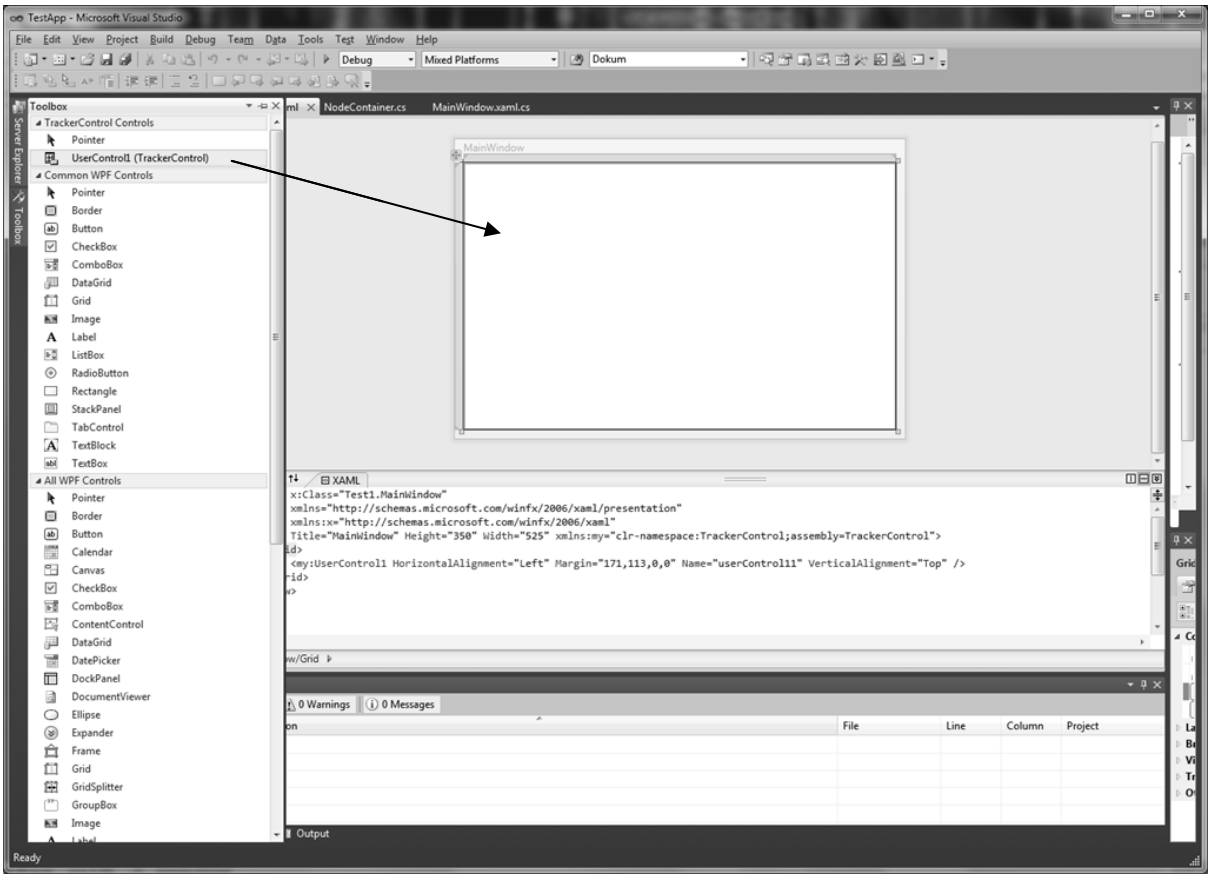
1. Definiera typerna
2. Definiera noderna
3. Definiera pilarna

6 Resultat och utvärdering

Detta kapitel behandlar prototypen av programmet. Det kommer att visa hur användarkontrollen används och övergripande vilka metoder som används bland annat för att skicka in data till den samt för att rita upp grafen.

6.1 Användandet av slutprodukten

Som nämnts i avsnitt 2.3 så skulle en användarkontroll som kan återanvändas i flera applikationer skapas. I Figur 6.1 visas det att spårbarhetslösningen dyker upp i Visual Studios verktygslåda och har namnet *TrackerControl*. En utvecklare kan dra in en TrackerControl i ett applikationsfönster för att på så vis använda sig av kontrollen.



Figur 6.1 Användarkontrollen dras in i ett huvudfönster

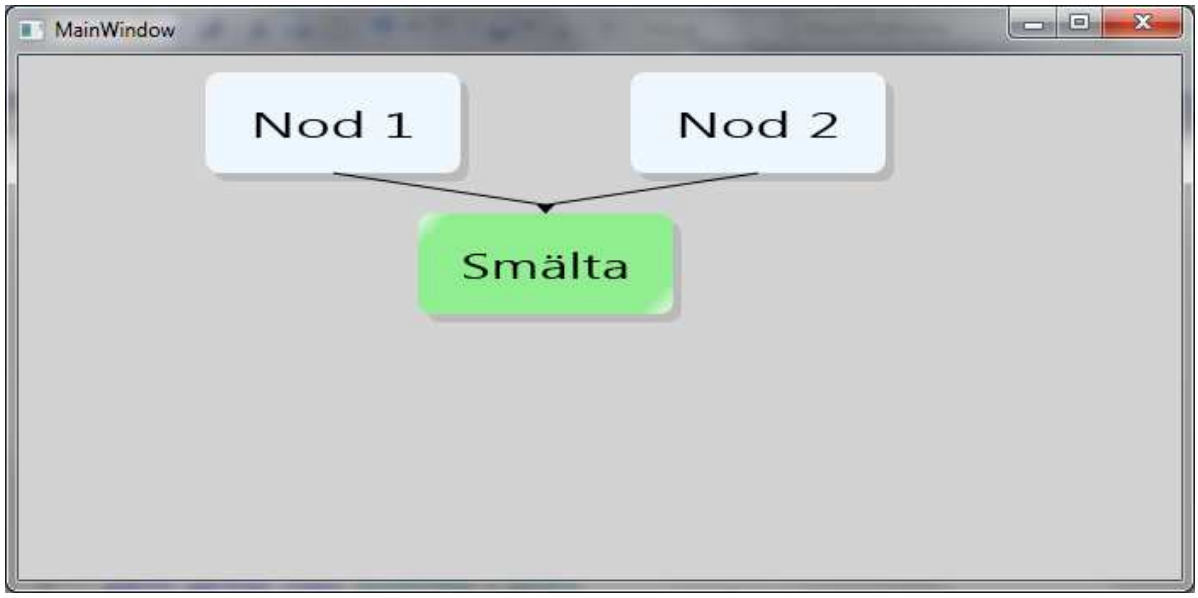
I Figur 6.2 skapas först två nodtyper som namnges *Smälta* och *Råvara*. Därefter läggs noder till med metoden AddNode() och pilar med AddConnector(). Efter detta så ritas grafen upp med metoden Draw(). För en mer utförlig beskrivning av metoderna se bilaga.

```
using TrackerControl;

namespace Test1
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            NodeType råvara = myTrackerControl.AddType("Råvara", Colors.AliceBlue);
            NodeType smälta = myTrackerControl.AddType("Smälta", Colors.LightGreen);
            myTrackerControl.AddNode(1, "Nod 1", 1, råvara);
            myTrackerControl.AddNode(2, "Nod 2", 1, råvara);
            myTrackerControl.AddNode(3, "Smälta", 2, smälta);
            myTrackerControl.AddConnector(1, 3, "Smältning");
            myTrackerControl.AddConnector(2, 3, "Smältning");
            myTrackerControl.Draw();
        }
    }
}
```

Figur 6.2 Exempel på kod

Resultatet av koden i Figur 6.2 visas i Figur 6.3.



Figur 6.3 Resultat av koden i Figur 6.2 Exempel på kod

Som det nämndes i Kapitel 5 så kan en XML-fil användas för att skicka in data om komponenterna i grafen. Detta görs med metoden ReadFromFile(String sökväg), nedan visas ett exempel på hur den kan användas.

```
trackerControl.ReadFromFile(@"D:\XMLFile.xml");
trackerControl.Draw();
```

Genom att anropa medoden Draw() ovan så ritas sedan grafen upp i fönstret.

För att spara en graf till en XML-fil anropas metoden WriteToFile(String sökväg). Denna funktion var inget som fanns med i specifikationen men var ändå något som ansågs nödvändigt att implementera för att applikationen senare skall kunna vidareutvecklas.

6.2 Ytterligare funktionalitet

Förutom den funktionalitet som diskuterats i Kapitel 4 och 5 så finns det funktioner som; zoomar in och ut grafen, returnerar en nod till applikationen som implementerar användarkontrollen, jämför gemensamma föregående noder för två noder. Detta avsnitt beskriver dessa kort.

6.2.1 Zoomningsfunktionen

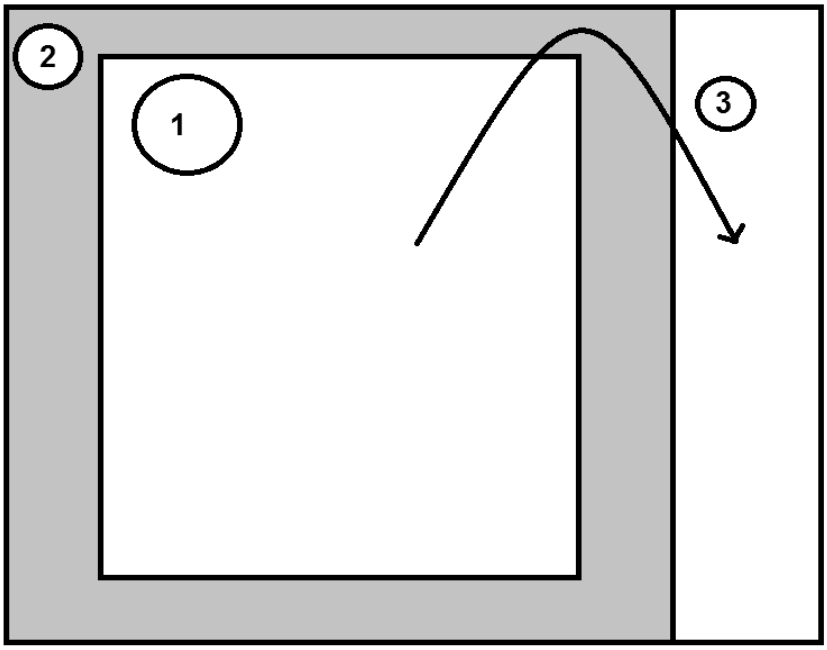
Användarkontrollen har stöd för att zooma in och ut grafen. Detta görs med metoden Zoom(ZoomModes) där ZoomModes är en enum [53] i användarkontrollen som kan anta värdena *In* och *Out*. Om ZoomModes.In skickas in i metoden så förstoras noderna med 5 % och om ZoomModes.Out skickas in så förminskas noderna med 5 %. Anledningen till att just 5 % valdes var för att det ansågs vara ett lagom stort tal för att inte få för stora hopp mellan storlekarna.

6.2.2 Presentation av nodernas information

De dynamiska informationsrutorna valdes att användas när användaren för muspekaren över en nod. Denna lösning presenterades i avsnitt 4.4.1. Idén att använda statiska informationsrutor (avsnitt 4.4.2) slopades. Detta då det strävas efter att endast presentera en interaktiv graf i användarkontrollen och inte allokera plats för en statisk informationsruta. Istället valdes det att implementera ett event, NodeClicked, som returnerar den nod som användaren klickar på till huvudapplikationen. Det är sedan användaren av användarkontrollen som får bestämma hur denna information ska presenteras. Detta beskrivs utförligare i 6.2.3.

6.2.3 Returnera en klickad nod

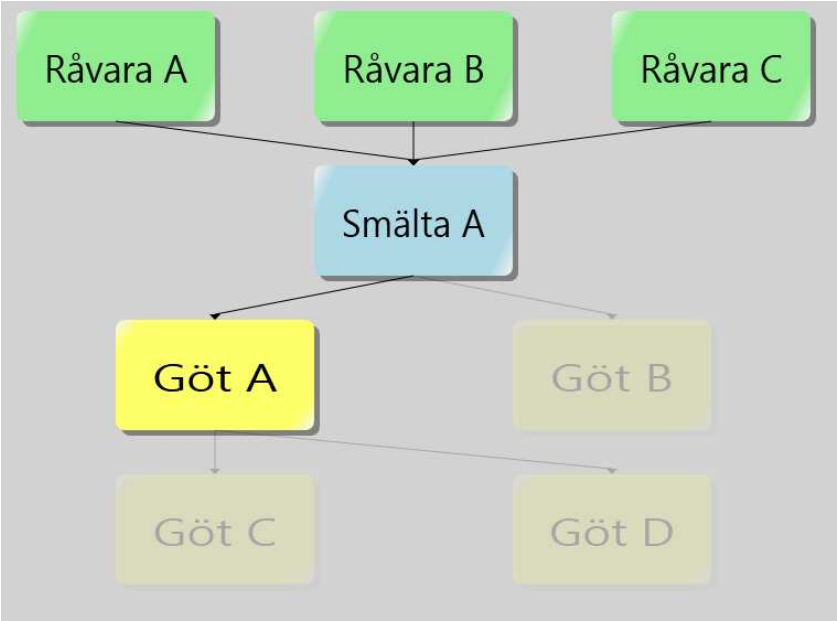
Användarkontrollen implementerar ett event kallat NodeClicked. Om huvudapplikationen, som implementerar användarkontrollen, fångar detta så returnerar användarkontrollen de noder användaren klickar på till huvudapplikationen. På så vis kan information om noderna fångas upp och presenteras utanför användarkontrollen. I prototypen i avsnitt 6.3 används detta event för att presentera information om noderna i en textruta utanför användarkontrollen. Figur 6.4 beskriver hur detta event skickar ut noden till huvudapplikationen som i sin tur skickar den till en textruta som presenterar informationen om noden.



Figur 6.4 Numrerade objekt är 1. Användarkontrollen 2. Huvudapplikationen 3. Textruta i huvudapplikationen

6.2.4 Visa gemensamma noder

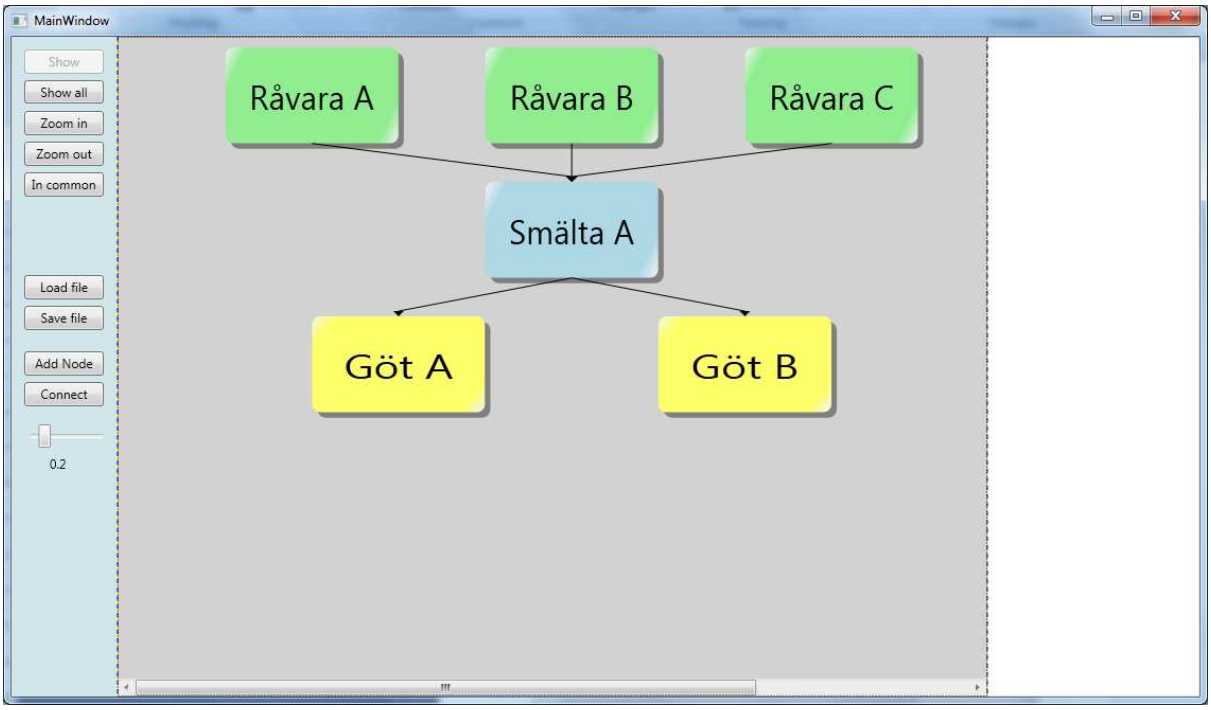
Genom att skicka in två noder i metoden InCommon(Node, Node) så kommer de noder som de två noderna gemensamt har sitt ursprung i att behålla sin färg, resten färgas gråa. Denna funktion kan vara praktisk om användaren till exempel vill veta eventuell felkälla från två göt av dålig kvalitet. Se Figur 6.5 för ett exempel. Denna funktion visas upp i protoypen.



Figur 6.5 Gemensamma ursprungsnoder för Göt C och Göt D behåller sin färg. Orelaterade blir genomskinliga, i detta fall Göt B.

6.3 Prototypen

För att visa användarkontrollens funktionalitet skapades en fristående applikation som implementerade den. Den innehåller endast enkel funktionalitet som visar hur data kan laddas in. I Figur 6.6 visas denna applikation. Den innehåller till vänster en meny med knappar för diverse funktioner, i mitten finns användarkontrollen och till höger en textruta som presenterar mer utförlig information om de noder som användaren klickar på. Textrutan får sin information genom att hämta data ur den nod som returneras av eventet NodeClicked som presenterades i avsnitt 6.2.3.



Figur 6.6 Prototypens utseende

6.4 Utvärdering av den tekniska designen

Vid syning av prioriteringslistan i avsnitt 2.1.3 så visar det sig att samtliga punkter har utförts. Grafen presenteras vertikalt och det är möjligt att färglägga de olika typerna av noder. Det är även möjligt att zooma in och ut grafen. Analysen av smältor kan antingen visas i den dynamiska informationsrutan eller presenteras utanför användarkontrollen genom att skicka ut noden med eventet NodeClicked. Anledningen till att det inte valdes att implementera en statiskt placerad informationsruta var att användarkontrollen primärt ska användas till att presentera en graf. Hur informationen om varje nod sedan presenteras är upp till användaren av användarkontrollen att avgöra. Lösningen är dessutom kostnadsfri utöver de utvecklingsverktyg som presenterades i avsnitt 2.4.

6.5 Utvärdering av grafiska designen

Användargränssnittet har fått ett hyfsat attraktivt utseende, det går säkerligen att förbättra men dock anses det fullt acceptabelt. Sett till de *gyllene reglerna* i avsnitt 3.1.1 så har det uppfyllts eller åtminstone getts stöd för att uppfylla regel #1, #3, #5 och #8. Programmet är på grund av sin enkelhet konsistent på det viset att alla komponenter i programmet har ett tydligt syfte och följer ett tydligt mönster. En informativ återkoppling ges dels med den dynamiska informationsrutan samt med eventet NodeClicked som tidigare diskuterades i kapitlet. God information leder också till en minskad användning av korttidsminnet. Då programmet är relativt enkelt i sin funktion så har antalet fel kunnat minimeras och skapat ett tydligt API.

En av punkterna i de gyllene reglerna som uppnåtts på grund av tidsbrist är #2, det vill säga att ge stöd för olika typer av användare. Troligtvis vore det väldigt nyttigt att ha ett läge för nya användare av programmet samt ett läge för avancerade användare som har använt programmet och är vana med dess funktioner. Det kan även tänkas att det kan finnas ett användarläge för administratörer.

6.6 Summering

Detta kapitel har presenterat hur prototypen kan användas genom att beskriva metoderna för att skicka in information om komponenterna i grafen. Några andra metoder som presenterats är hur användaren kan zooma in och ut grafen samt hur två noders gemensamma ursprung kan visas. Eventet NodeClicked har presenterats vilket returnerar noden som användaren klickar på så dess information kan presenteras utanför användarkontrollen.

7 Slutsats

Syftet och målet med projektet var att skapa en användarkontroll som ritar ut en spårbarhetsgraf från inskickad data enligt principen som beskrivs i Figur 1.2. Den skulle även vara användarvänlig samt ha en attraktiv design. Projektets resultat speglar de uppsatta målen då projektets krav har uppfyllts. Vi har främst fördjupat våra kunskaper inom .NET, då främst Visual C# och WPF, genom att göra detta arbete och vi kommer att diskutera examensarbetets för- och nackdelar i detta kapitel. Förutom det så kommer även detta kapitel sammanfatta uppsatsen genom att diskutera de krav vi har uppfyllt.

7.1 Uppfyllda mål

Nedanstående krav är uppfyllda. Vissa funktioner har implementerats tämligen enkelt medan vissa önskemål har studerats innan användning.

- Applikationen visar en graf med noder och pilar. Grafen presenteras på ett centrerat vis och vertikalt istället för horisontellt.
- Grafens komponenter sparas och läses in från en extern källa i form av en XML-fil.
- När en nod klickas på får användaren information om denna. Lösningen för detta är dels en dynamisk informationsruta samt att noden genom ett event skickas ut ur användarkontrollen till huvudapplikationen. På så vis kan programmet som implementerar användarkontrollen extrahera information ur noden som skickats in i användarkontrollen och presentera det som eftersöks. Avsnitt 6.2.3 förklarar detta mer ingående.
- Grafen har en användarvänlig struktur och presenterar information på ett förståndigt vis.
- Det går att följa en nod framåt och bakåt i grafen.
- Det är möjligt att förstora och förminska grafen för att åskådliggöra relevant information.
- Det är möjligt att presentera två noders gemensamma ursprung (se Figur 1.4) genom att skicka in dessa i användarkontrollen. Se 6.2.4 för mer information.

- Det grafiska användargränssnittet är rent utseendemässigt tilltalande.
- Den dynamiska informationsrutan visar grundläggande information om komponenterna i användarkontrollen på ett snabbt och bra vis.

Anledningen till att vi har löst vissa önskemål på särskilda sätt har diskuterats i mer ingående i Kapitel 3 och 4. Vi har till exempel presenterat grafen centrerad för att kortare sackader leder till större användarvänlighet.

7.2 Vidareutveckling

Vidareutveckling av applikationer är alltid välkomnat då det kan läggas till ny funktionalitet, fixa eventuella fel och även hålla det grafiska användargränssnittet modernt. En självklar men omfattande vidareutvecklingsprocess är att slå ihop alla tre system som nämndes i avsnitt 2.1.1. Vi har i stor mån försökt göra vårt program utökningsbart genom att dela upp samtliga delar i programmet så mycket som möjligt. Som vi nämnt i avsnitt 6.5 ser vi även nyttan att utveckla en vy för en nybörjare samt en vy för en avancerad användare av programmet.

7.3 Utvärdering

Det har varit en stor fördel att utföra examensarbetet ute hos ett företag. Det har dels varit en mycket trevlig upplevelse rent socialt då en mycket trevlig stämning råder på kontoret, dels varit mycket nyttigt att kunna använda de anställdas kunskaper för arbetet.

En nackdel har varit att vi aldrig har haft kontakt med slutkunden, det vill säga den kund som Sogeti jobbar mot. Detta då det i slutändan är de som kan ha användning av programmet. Istället har vi använt oss av Sogeti som kund.

Om vi hade gjort om arbetet idag, med de kunskaperna vi har nu, hade vi inte spenderat lika mycket tid på att söka information om spårbarhet. Vi fann inga modeller för hur vi på bästa

sätt kan bygga en spårbarhetslösning. Därför borde vi ha börjat utveckla en egen idé i ett tidigare stadie.

I början av arbetet lades också mycket tid på att förbättra kunskaperna inom XAML, något som inte visade sig vara till någon större nytta. Detta då en mycket liten del av programmet består av kod i XAML, i stort sett bara utseendet för demonstrationsprogrammet. I huvudsak är koden skriven i C# vilket lämpar sig för mer dynamiska användargränssnitt likt det vi har utvecklat.

7.4 Slutord

Examensarbetet har resulterat i en applikation som möter de uppsatta målen från Kapitel 1. Därmed anses projektet ha ett lyckat utfall. Det har diskuterats hur applikationen kan vidareutvecklas och förbättras i detta kapitel. Nya kunskaper inom examensarbetets hårda samt mjuka delar har erhållits under arbetets gång. De hårda delarna anses vara verktygen och implementering av funktioner medans mjuka delar är exempelvis design för grafiska användargränssitt.

Vår handledare på Sogeti, Thomas Heder, tycker att applikationen lever upp till de utsatta målen och han tror att det kan komma att användas i andra projekt.

Referenser

1. Analogi. Nationalencyklopedin. [Online] [Citat: den 27 april 2011.] http://www.ne.se/analogi
2. Aluminium. Wikipedia. [Online] [Citat: den 21 februari 2011.] http://sv.wikipedia.org/wiki/Aluminium
3. Computer Sweden. [Online] [Citat: den 12 april 2011.] http://cstjanster.idg.se/sprakwebben/ord.asp?ord=informations%F6verbelastning
4. K omplementfärg. Wikipedia. [Online] [Citat: den 16 februari 2011.] http://sv.wikipedia.org/wiki/Komplementf%C3%A4rg
5. Mnemoteknik. Wikipedia. [Online] [Citat: den 26 april 2011.] http://sv.wikipedia.org/wiki/Mnemoteknik
6. Nervsystemet. Sahlgrenska. [Online] [Citat: den 25 april 2011.] http://cns.sahlgrenska.gu.se/goude/nsd/structure_992
7. Företagspresentation. Sogeti Sverige AB. [Online] [Citat: den 9 maj 2011.] http://www.sogeti.se/Om-Sogeti/Foretagspresentation/
8. Grundämnen. Wikipedia. [Online] [Citat: den 21 februari 2011.] http://sv.wikipedia.org/wiki/Lista_%C3%B6ver_grund%C3%A4mnen
9. Spårbarhet. Wikipedia. [Online] [Citat: den 05 april 2011.] http://sv.wikipedia.org/wiki/Sp%C3%A5rbarhet
10. Spårbarhetslösningar. SYSteam AB. [Online] [Citat: den 14 mars 2011.] http://www.systeam.se/sv/Vara-erbjudanden/PLMProduktutveckling/Sparbarhetslosningar/
11. Spårbarhet inom läkemedelsindustrin. [Online] [Citat: den 14 mars 2011.] http://www.bilsweden.se/BinaryLoader.axd?OwnerID=e8fac5ee-9083-48bf-b3af-fe2c5de49d42&OwnerType=0&PropertyName=File1&FileName=Conny%20Axelsson.pdf
12. Från jord till bord - Spårbarhet i livsmedelsbranschen. Lawson. [Online] [Citat: den 14 mars 2011.] http://scripts.cordeo.net/lawson/get_lores2.php?file=LAWWP16_FJTGESVA4_1008-
13. Gotel, O.C.Z. och Finkelstein, C.W. An analysis of the requirements traceability problem. u.o. : Colorado Springs, 1994.

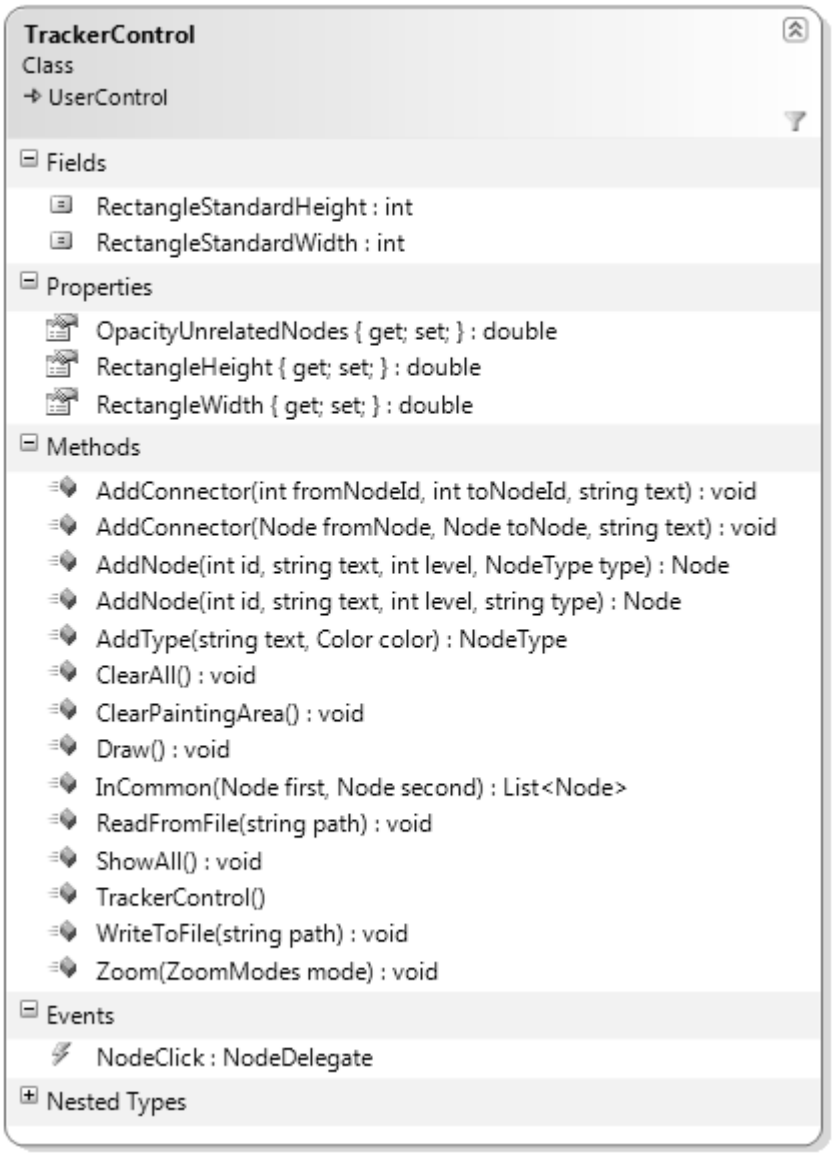
14. Application Programming Interface. Wikipedia. [Online] [Citat: den 16 maj 2011.] http://sv.wikipedia.org/wiki/Application_Programming_Interface
15. Visual Studio. Wikipedia. [Online] [Citat: den 28 februari 2011.] http://en.wikipedia.org/wiki/Microsoft_Visual_Studio
16. Windows. Wikipedia. [Online] [Citat: den 28 februari 2011.] http://en.wikipedia.org/wiki/Windows
17. Common Language Runtime. Wikipedia. [Online] [Citat: den 28 februari 2011.] http://en.wikipedia.org/wiki/Common_Language_Runtime
18. Om .Net Framework. Microsoft TechNet. [Online] [Citat: den 15 februari 2011.] http://technet.microsoft.com/sv-se/library/cc775788%28WS.10%29.aspx
19. Introduction to WPF. MSDN. [Online] Microsoft. [Citat: den 16 februari 2011.] http://msdn.microsoft.com/sv-se/library/aa970268.aspx
20. XAML. Wikipedia. [Online] [Citat: den 28 februari 2011.] http://en.wikipedia.org/wiki/Xaml
21. Märkspråk. Wikipedia. [Online] [Citat: den 28 februari 2011.] http://sv.wikipedia.org/wiki/M%C3%A4rkspr%C3%A5k
22. ASP.NET. Wikipedia. [Online] [Citat: den 28 februari 2011.] http://en.wikipedia.org/wiki/ASP.NET
23. Web service. Wikipedia. [Online] [Citat: den 3 mars 2011.] http://sv.wikipedia.org/wiki/Web_service
24. Visual Basic. Wikipedia. [Online] [Citat: den 28 februari 2011.] http://en.wikipedia.org/wiki/Visual_basic
25. Visual C#. MSDN. [Online] [Citat: den 28 februari 2011.] http://msdn.microsoft.com/en-us/library/kx37x362.aspx
26. Visual C++. MSDN. [Online] [Citat: den 28 februari 2011.] http://msdn.microsoft.com/en-us/library/60k1461a.aspx
27. XML. Wikipedia. [Online] [Citat: den 16 maj 2011.] http://sv.wikipedia.org/wiki/XML
28. UTF-8. Wikipedia. [Online] [Citat: den 4 maj 2011.] http://sv.wikipedia.org/wiki/UTF-8
29. Microsoft Automatic Graph Layout. Microsoft. [Online] [Citat: den 16 maj 2011.] http://research.microsoft.com/en-us/projects/msagl/default.aspx

30. Microsoft Research. [Online] [Citat: den 16 april 2011.] http://research.microsoft.com/en-us/projects/msagl/195f1b23116b4f049b6e5dc815d96c89.png
31. Microsoft Store Online. [Online] [Citat: den 24 april 2011.] http://store.microsoft.com/microsoft/Automatic-Graph-Layout-2007/product/4DD40C40
32. Grapviz. Graphviz. [Online] den 26 april 2011. http://www.graphviz.org/
33. LionShare|Graphviz. Graphviz. [Online] [Citat: den 16 maj 2011.] http://www.graphviz.org/content/lion_share
34. Gränssnitt. Wikipedia. [Online] [Citat: den 27 april 2011.] http://sv.wikipedia.org/wiki/Gr%C3%A4nssnitt
35. Användargränssnitt. Wikipedia. [Online] [Citat: den 27 april 2011.] http://sv.wikipedia.org/wiki/Anv%C3%A4ndargr%C3%A4nssnitt
36. Graphical user interface. Wikipedia. [Online] [Citat: den 27 april 2011.] http://en.wikipedia.org/wiki/Graphical_user_interface
37. Grafiskt användargränssnitt. Wikipedia. [Online] [Citat: den 16 maj 2011.] http://sv.wikipedia.org/wiki/Grafiskt_anv%C3%A4ndargr%C3%A4nssnitt
38. [bokförf.] Ben Schneiderman och Catherine Plaisant. Designing the user interface. u.o. : Pearson Education, Inc., 2005, ss. 74-75.
39. Locus of control. Wikipedia. [Online] [Citat: den 28 april 2011.] http://en.wikipedia.org/wiki/Locus_of_control
40. Varför läser vi från vänster till höger? Illustrerad Vetenskap. [Online] den 14 mars 2011. http://illvet.se/fraga-oss/varfor-laser-vi-fran-vanster-till-hoger
41. Eye movements for pictures. Answers.com. [Online] [Citat: den 15 april 2011.] http://www.answers.com/topic/eye-movements-for-pictures
42. Eye movement in language reading. Wikipedia. [Online] [Citat: den 15 april 2011.] http://en.wikipedia.org/wiki/Eye_movement_in_reading
43. Shneiderman, Ben och Plaisant, Catherine. Designing The User Interface. International Edition. 4:e upplagan. u.o. : Pearson Education, Inc., 2005. ss. 510-512.
44. Benyon, David och Turner, Phil och Turner, Susan. Designing Interactive Systems. u.o. : Pearson Education Limited, 2005. s. 32.

45. Ceesay, Lamin och Hammam, Georges. Empiriskt grundade designprinciper för användarvänliga gränssnitt. En studie med fokus på biljettautomatsystem. [Online] [Citat: den 21 mars 2011.] http://bada.hb.se/handle/2320/6482
46. Tidwell, Jenifer. Designing Interfaces. u.o. : O'Reilly Media, Inc, 2006. s. 280.
47. Warm VS. Cool Colors. Feng Shui And Beyond. [Online] [Citat: den 8 april 2011.] http://www.feng-shui-and-beyond.com/color-psychology-warm.html
48. Varningsmärken och skyltar. PPV.se. [Online] [Citat: den 25 april 2011.] http://www.ppv.se/V%C3%A4gm%C3%A4rkenochskyltar/AVarningsm%C3%A4rken/tabid/69/language/sv-SE/Default.aspx
49. Ordförklaringar. Pinnacle Studio. [Online] [Citat: den 22 mars 2011.] http://pinnaclestudio.helpmax.net/sw/ordforklaringar/
50. Figurerna som lurar hjärnan. Allt om vetenskap. [Online] [Citat: den 4 maj 2011.] http://www.alltomvetenskap.se/index.aspx?article=396
51. Colors Class (System.Windows.Media). MSDN. [Online] [Citat: den 2 maj 2011.] http://msdn.microsoft.com/en-us/library/system.windows.media.colors.aspx
52. Komplementfärger. Wikipedia. [Online] [Citat: den 12 juni 2011.] http://sv.wikipedia.org/wiki/Komplementf%C3%A4rg
53. enum (C#). MSDN. [Online] [Citat: den 26 maj 2011.] http://msdn.microsoft.com/en-us/library/sbbt4032%28v=vs.80%29.aspx

Bilaga 1 API mot användarkontrollen

Gränssnittet mot användarkontrollen visas nedan.



Egenskaper för TrackerControl

OpacityUnrelatedNodes
Opacitetsvärdet för gömda noder.

RectangleHeight
Nodernas höjd.

RectangleWidth
Nodernas bredd.

Metoder för trackerControl

AddConnector()
Lägg till en pil mellan två noder.

AddNode()
Lägg till en nod i grafen.

AddType()
Lägg till en nodtyp (till exempel smälta, råvara eller göt).

ClearAll()
Ta bort alla komponenter i användarkontrollen och ta bort den uppritade grafen.

ClearPaintingArea()
Ta bort den uppritade grafen, komponenterna finns dock kvar i grafen.

Draw()
Rita upp en spårbarhetsgraf med hjälp av de noder och pilar som skickats in i användarkontrollen.

Incommon()
Dölj de noder som de två inskickade noderna inte har gemensamt.

ReadFromFile()
Läs in noder och pilar från en inskickad XML-fil.

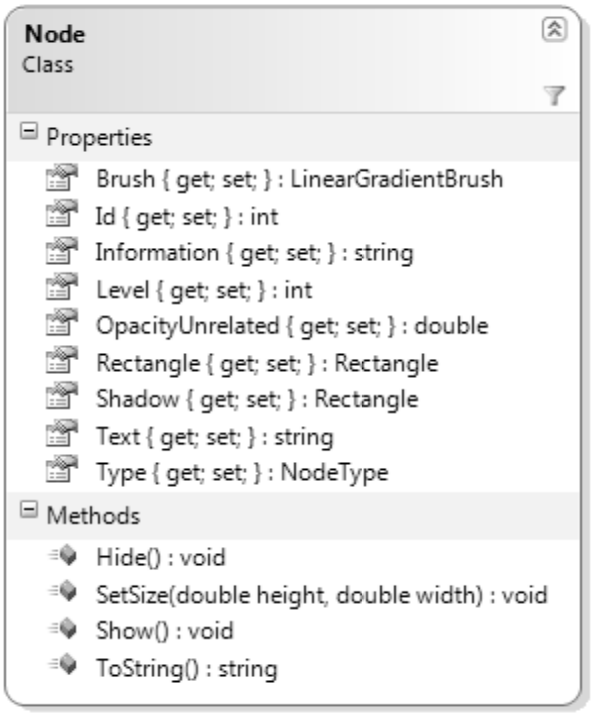
ShowAll()
Visa alla noder i sin ursprungsfärg, det vill saga om en nod är dold så visas den igen.

WriteToFile()
Skriv den uppritade grafen till en XML-fil med den angivna sökvägen.

Zoom()
Zooma in eller ut ur grafen. ZoomModes.In zoomar in, ZoomModes.Out zoomar ut grafen.

Eventet NodeClick
Eventet som fångas upp i huvudapplikationen för att ta emot noder som användaren klickar på.

Gränssnittet mot noden visas nedan.



Egenskaper för Node

Brush
Ställer in den pensel som ska användas för att färglägga en nod.

Id
Nodens unika id.

Information
Utökad information om en nod.

Level
Den rad noden ska placeras på.

OpacityUnrelated
Ställer in vilken opacitetsgrad som gömda noder ska anta.

Rectangle
Den rektangel som noden består av.

Shadow
Nodens skugga i form av en rektangel.

Text
Nodens synliga text.

Type
Nodens typ.

Metoder för Node

Hide()
Göm noden, det vill säga minska dess opacitet.

SetSize()
Ändra höjden och bredden på noden.

Show()
Visa noden, det vill säga vissa med opacitetsvärdet 1.

ToString()
Returnerar information om noden. Det vill säga dess information, namn och ämnestyp.