Användargränssnitt för HAAS CNC maskiner

LUNDS
UNIVERSITET
Lunds Tekniska Högskola

LTH Ingenjörshögskolan vid Campus Helsingborg
Institutionen för Datavetenskap

Examensarbete:
William Truong
Sasa Perak

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

Tryckt i Sverige
Media-Tryck
Biblioteksdirektionen
Lunds universitet
Lund 2009

Sammanfattning

Användargränssnitt för HAAS CNC maskiner

Vårt arbete är baserat på att göra ett användargränssnitt för CNC (Computer Numerical Control) maskiner tillverkat av HAAS. CNC maskinen använder sig av G-Kod programmering som för den vanlige metallsvarvaren är svårt och jobbigt att lära sig. Detta avskräcker många svarvare, som använt manuella maskiner, att gå över till CNC.

Vår uppgift är att göra övergången från manuella till automatiska maskiner enkelt. Vi och vår kund tycker att vi har uppnått detta genom att göra vårt användargränssnitt enkel men ändå avancerad nog för att kunna göra komplexa objekt. Eftersom programmet är gjort efter instruktioner på hur en svarvare vill att programmet ska vara uppbyggt, så är det snabbt förståeligt för andra svarvare.

Efter noggrant arbete anser vi att vårt program täcker det mesta av vad en svarvare skulle vilja framställa. Vid behov av tillämpningar är vårt program fullt utrustad för att klara av tillägg av funktioner.

Nyckelord: CNC, G-Kod, GUI, HAAS, Användargränssnitt

Abstract

User Interface for CNC HAAS machines

Our work is based on making a User Interface for CNC (Computer Numerical Control) machines manufactured by HAAS. The machine uses a programming language called G-Code, which for the ordinary lathe-operator is difficult to learn. This discourages many operators, who have only used manual machines, to convert to CNC.

Our task is to make the transition from manual to automatic machines easy. We and our client deems that we have achieved this by making our User Interface simple but advanced enough to make complex objects. Since the program is designed after instructions by how a lathe-operator wants the program to function, so that it is quickly understandable by other operators.

After careful development of our work, we consider our program to cover most of what a lathe operator would produce. In the evens of possible additions, our program is fully compatible to handle added functions.

Keywords: CNC, G-Code, GUI, HAAS, User Interface

Förord

Kurser avklarade, tentamen avklarade…
Det som återstod var ett examensarbete som skulle passa två personer, som varit dataintresserade sen barndomen. Det som startade glöden skulle kunna sammanfattas med ett enda ord, NES (Nintendo Entertainment Systems) [1]. Denna glöd har varit en välsignelse samt en förbannelse. En förbannelse som fått oss att sitta uppe natten lång för att klara av den sista nivån, fast också en välsignelse för att ha gett oss den lyst som krävs av varenda högpresterande programmerare.

Vi började fråga oss själva: ”Hur har detta programmerats?”. Denna fråga ledde oss till Lunds Tekniska Högskola där vi sökte till att utbildas till framtida dataingenjörer. Vår kunskap inom området ökades i takt med jakten efter svårare utmaningar. Denna jakt har lett oss till företaget, Legoteknik AB, där vi beslutade att använda våra kunskaper.

Legoteknik AB är ett metallsvarvningsföretag där de nyligen gått över till att använda CNC (Computer Numerical Control) [2] maskiner. Dessa maskiner är datorstyrda och utför arbetet beroende på värden som matas in av användaren. Vissa av dessa maskiner använder sig av G-Kod[10] programmering, vilket för många av arbetarna är svårt att lära sig. Vi fick uppgiften att göra denna G-Kod programmering lätt att använda. Vi kom fram till att göra ett användargränssnitt där en svarvare som enbart använt sig av manuella svarvningsmetoder, ska ha inga problem att använda en CNC maskin som kräver stor kunskap inom G-Kod programmering.

Med en av företagets mest kunniga person med full kunskap över G-Kod och alla aspekter inom svarvning, som vår handledare, var vi redo att skapa vårt gränssnitt som skulle vara lätt att använda för en svarvare.

Det var stunder inom projektet som fick oss att ifrågasätta vår kunskap inom programmering, men med ihärdighet och tålamod kom vi alltid på en bra lösning som skulle lösa och förbättra en funktion.

Innehållsförteckning

1 Inledning

Det mesta i tillverkningsindustrin styrs idag av CNC (Computer Numerical Control) maskiner. Principen med CNC är att man matar in G-Koder för att styra ett verktyg t.ex. borr, svarvare till att producera objekt från material som aluminium, stål, mässing osv. Dessa objekt kan vara allt från kretskort till avancerat formade stålkonstruktioner.

CNC maskinen vi ska jobba med är gjort av företaget HAAS. Dem själva gör inget grafiskt användargränssnitt (GUI) till maskinen utan det finns endast tredjeparts program ute i marknaden. Dessa program passar inte ihop med kundens önskemål på grund av olika begränsningar.

1.1 Bakgrund

Föregångaren till CNC maskinen är NC[3](Numerically Controlled) som utvecklades av John T. Parsons slut 1940-talet. Dessa maskiner var ”hårdvaruprogrammerade”, vilket betydde att dem var utvecklade till att utföra en specifik uppgift och var inte särskild flexibla för ändringar. I början användes hålkort som medium fram tills den ersattes av disketter och serieportkablar. Språket som användes här var G-Kod, vilket överfördes vidare till CNC.

Även om man än idag kan köra CNC maskiner genom att skriva i ren G-Kod så används det sällan. Dem flesta idag använder CAM[4] program (Computer-Aided Manufacturing). Det är ett programmeringsverktyg där man ritar upp designen på produkten med 3D dvs. en typ av CAD[5] program. Sedan generar programmet den färdiga G-Koden.

1.2 Syfte

Syftet med programmet är att ge oss en djupare kunskap inom programmering där vad vi får göra är begränsat till kundens önskemål.
Programmet ska vara ett användargränssnitt för att göra det möjligt för en vanlig svarvare utan kunskap inom G-Kod programmering, att kunna framställa den produkt som önskas.

Av vad vi fått veta av kunden finns inget program som vårt ute på marknaden som är användarvänligt för den som inte kan G-Kod. Det finns dessutom CAM program som hanterar CNC maskiner, fast dessa program är väldigt dyra samt väldigt svårt att hantera för den vanlige svarvaren. Vi tror starkt på att vårt program kommer effektivisera produktionen inom företaget.

Programmet kan bli väldigt stort, så för tillfället håller vi oss till följande avgränsningar:

- *Programmet ska kunna hantera de flesta ritningar*
- *Kan endast köras på HAAS maskiner*
- *Språken; Engelska och Svenska*

1.3 Problemformulering

Under projektets gång måste vi följa ett par viktiga frågor som måste hållas för att slutprodukten blir det vi siktat på.

- *”Är programmet lätthanterligt?”*
- *”Kommer svarvaren att kunna anpassa sig?”*
- *”Kommer programmet att användas?”*

De är vår guide, våra riktlinjer, och grundprincipen till programmet.

2 Arbetsmetoder

Val av programmeringsspråk som skulle användas baserades på vår tidigare utbildning i skolan. Java var det första vi tänkte på, men vi utforskade möjligheter inom C ++. Det visade sig att C ++ är alldeles för annorlunda från Java och eftersom vi har haft mycket mer utbildning i Java, bestämde vi oss för att skriva programmet i J#.

Vi använde oss av Microsoft Visual J# .Net Framework [6] eftersom Microsofts Framework hanterar textrutor och knappar på ett snabbt och effektivt sätt.

Vi planerade först att ha krav- och designspecifikationer[bil] klart uppsatta innan vi började med programmeringen, fast eftersom kunden inte kunde fastställa vad exakt han ville ha, var vi tvungna att improvisera och programmera efter kundens önskemål vid olika tillfällen.

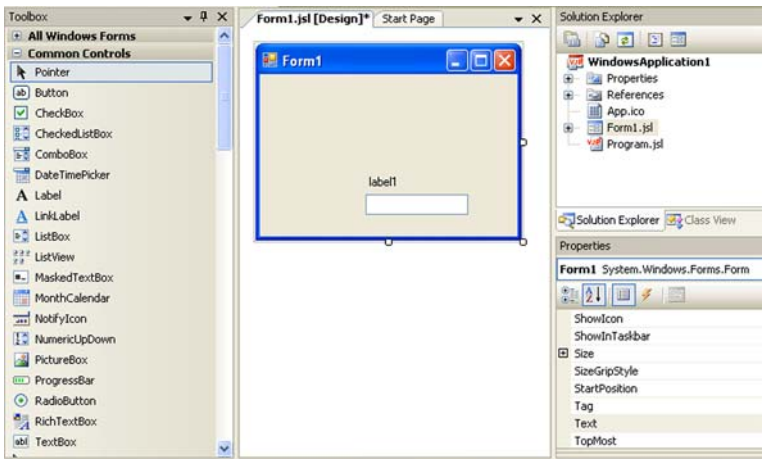
Arbetsgången blev att vi gjorde en funktion och design efter att ha haft ett möte med kunden där han lämnade sina synpunkter och eventuella ändringar som behövdes göra. Efter varje möte hade vi ingen kontakt med kunden, såvida vi inte hade några frågor, tills nästa möte där vi skulle få reda på vad han senare ville ha.

I projektets slutfas då alla de stora funktionerna har lagts dit, skapade vi en beta-version av programmet där kunden fick programmet för att testa med maskinerna. Eventuella små fel fixades efter ett snabbt telefonsamtal varav vi skickade kunden en ny version.

Vid kompilering av programmet använde vi först Visual J# egen publicerings-funktion, men det hade för lite valmöjligheter samt att den packade inte med alla nödvändiga filer för att starta vårt program i en helt ny data. Vi utforskade andra möjligheter att publicera vårt program, vilket ledde oss till Installshield[7]. Detta program visade sig vara väldigt kraftfullt med många inställningar, men också skapade många problem med att kompilera en installationsfil.

2.1 IDE

Utvecklingsmiljön(Integrated Development Environment) vi jobbade med var Visual J#, som vi nämnt innan. Programmet är väldigt lätt att sätta sig in i för nybörjare och är ganska användarvänlig.



Principen är enkel. Fönstret i mitten på bilden ovan visar en arbetsyta (design), s.k. GUI, varav vi har lagt dit en etikett och en textruta. Man skapar dessa genom att välja bland dem olika former av objekt som finns på menyn till vänster om arbetsytan. När du lägger dit ett objekt på arbetsytan så generar Visual J# koden automatiskt och lägger det i ditt källkodsfönster. Om du sen tar bort objektet från arbetsytan så rensar programmet också koden för objektet i din källkod.

Till höger om arbetsytan har du bl.a. egenskaper (properties) på objekt när du markerar dem. Där kan du ändra allt från storlek, namn på objektet, lägga dit händelser etc. Ovanför egenskaper har du en översikt på projektet. Där kan du se vilka filer som finns i projektet, ändra namn på dem, lägga dit fler filer etc.

Förutom arbetsytan kan du också titta på programmet i källkod också. Du kan lägga dit objekt, ändra egenskaper etc. med källkod men då måste du skriva allt själv, vilket är tidskrävande. Källkoden används därför till andra uppgifter som t.ex. skriva kod till olika händelser (events).

Själva kompileringen är inget speciellt utöver andra. Den listar fel den hittar, du kan klicka på felen och komma dit direkt i källkoden. Om du fastnar på kompileringen så finns det hjälpfunktioner i programmet som förklarar generellt vad felet är.

2.2 Backup

Metoder som användes var:

- USB Minnen
- Google Internetlagring
- Stationär/Bärbar Dator

Efter vad vi tyckte var en stor ändring, gjorde vi dessutom en extra backup av hela projektet, s.k. *draft*, så vi har en punkt att gå tillbaka till vid framtida möjliga missöden. Fördelen av att ha gjort en massa drafts gjorde det möjligt för oss att titta tillbaka till olika delar under projektets gång för att se hur våra tankesätt om programmet sakta ändrades till det som är idag.

2.3 Dokumentation

Under varje möte med kunden antecknades allt. Illustrationer ritades av kunden för hand för att förklara hur han ville att en viss funktion skulle röra sig och hur dess motsvarande G-Kod skulle skrivas ut till följd av en viss programmerings-sekvens.

Rapporten uppdaterades varje gång framgång gjordes i projektet. Denna rapport var en länk mellan oss och vår examinator som skulle övervaka vårt projekt. Detta tillsammans med våra drafts, gav oss tillräckligt med information för att kunna se hur snabbt projektet fortskridit samt hur vi utvecklats tankemässigt.

2.4 Tidsplanering

Vår tidsplanering [Bilaga 6.2.5] blev inte som vi ursprungligen planerat, dels p.g.a. programmeringen var mycket större än vad vi ursprungligen förutspått, samt att vid enstaka tillfällen kom privatlivet i vägen också.

Från början trodde vi att programmet skulle bli lätt att lösa eftersom vi körde med ett program som underlättade GUI:n för oss samt att vi körde på ett språk vi kändes oss trygga med, men vi fastnade ett par gånger som tog tid att lösa. Detta var mycket p.g.a. vår erfarenhet inom språket samt att kunden kom oftast med nya idéer som han ville ha med.

Kunde vi ha gjort programmet färdigt snabbare än i nuläget? Ja och nej, det fanns rum för att jobba mer på programmeringen, men eftersom kunden hade problem att ge oss en helhetsbild kändes det mer logiskt för oss att göra dem önskemål kunden kom på i efterhand först och checka av dem innan vi rusha fram till nästa grej direkt. Det blev bara mer jobb i form av ändringar och tillägg om vi försökte göra detta.

3 Resultat

Innan projektet började, ville kunden att projektet skulle vara lätt att hantera för den vanlige svarvaren. Hans resonemang att de program som för tillfället fanns ute på marknaden var alldeles för begränsade och svårhanterliga, som oftast krävde en tekniker, från tillverkaren, att hjälpa till.

Efter att ha testat vårt program grundligt konstaterade kunden att detta är ett väldigt smidigt och bra program som kommer att göra det lätt för varje svarvare att använda sig av CNC maskiner.

Programmet har ansetts vara tillräckligt lätt att lära sig, att en svarvare med någon timmes genomgång av programmet ska kunna använda programmet till dess fulla potential.

Vid möjliga fel då värden som matas in är inkorrekta, finns en detaljerad *felsökningsfunktion* som visar för användaren var denne har skrivit fel. CNC maskinen har dessutom själv en grafisk simulator där användaren kan kolla att koden, som vårt program genererat, utför det som användaren önskat.

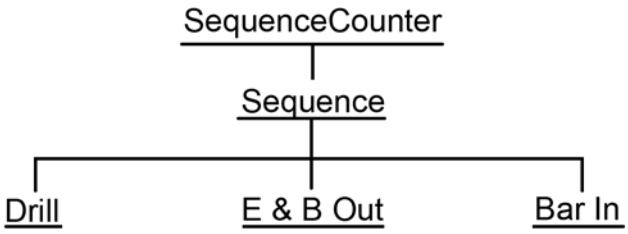
Det som tog mestadels av vår tid vid programmeringen var funktioner och operationer bakom själva GUI:n. Eftersom vi använde Visual Basic J# var det aldrig några stora problem att snabbt ändra UI om kunden kom med nya önskemål eller ville ha något ändrat. Det ändrar ändå inte dess faktum att GUI:n var det viktigaste biten, i alla fall för kundens sida. Kunden hade från början en ganska klar bild på hur han ville ha uppbyggnaden på GUI. Dem hade ett annat CNC maskin i företaget som dem köpt innan HAAS maskinen som hade ett riktigt bra GUI. Han ville ha ungefär samma bygge som den.

Grundprincipen för kundens GUI var att man fyller i nödvändiga parametrar från vänster till höger, sen hoppar man till en ny rad med nya parametrar. Man ser helt enkelt en liten bit i taget t.ex. man börjar först med att döpa vad program filen ska heta, vilket material du ska jobba på dvs. generella sakerna först innan man går djupare in.

Vi tänkte hela tiden på detta när vi byggde vårt program. Det fanns dock tillfällen där vi avvek från detta om vi kom på ett eget förslag och kunden tyckte det var en ännu bättre lösning.

3.1 Klasser

Klasstrukturen är följande:



Högst upp har vi SequenceCounter. Den klassen hanterar alla sequence-objekt som skapas inne i vår sequence-vektor. Här flyttas, tas bort, ändras övriga sequence-objekt.

I Sequence klassen bestäms vilken typ av objekt det ska vara t.ex. Drill, Bar In. Objektet och UI:n skapas och läggs in i SequenceCounter. Under Sequence finns varje typ av objekt. Inne i dessa klasser definieras UI:n.

Specifikationerna för klasserna finns i bilaga 6.1.

3.2 GUI Design

Vi har delat in vår design in olika kapitel, ”Revision”, och varje kapitel har dess versionsnummer, ”Draft”. Varje revision innehöll en stor ändring inom programmet, både kodningsmässigt samt

3.2.1 Revision 1: Draft 1

Vårt första försök på GUI:n var enkelt byggd, mestadels löst baserad på hur kunden ville ha programmet.

Kunden gjorde det klart att han ville ha en GUI med stegvis användarinput. Vid detta tillfälle hade ingen manuell J# programmering påbörjats, utan allt har gjorts med Visual J# Framework. Denna första ”draft”, som vi kallar det, gav oss kunskap om hur programmet fungerar med dess textbox deklarationer m.m.

3.2.2 Revision 2: Draft 1

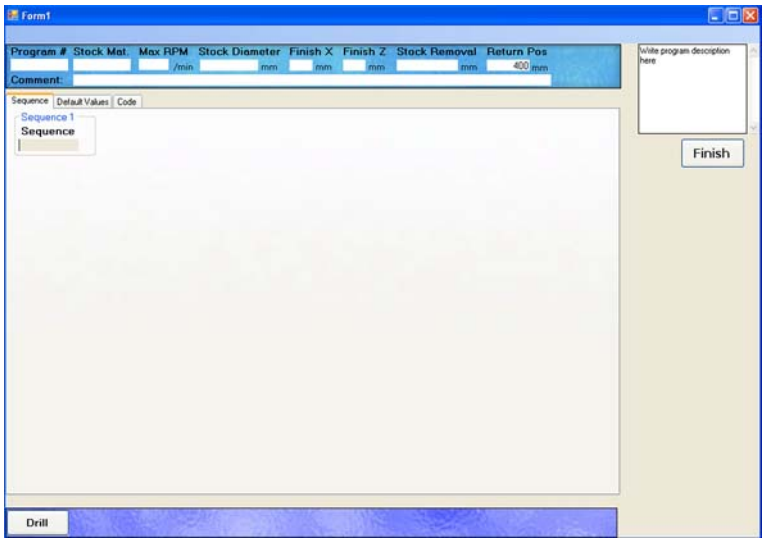
Efter att ha visat principen till GUI:n för kunden, arbetade vi vidare på att förfina designen med ordentligt placerade textboxar.

Alla ändringar i detta stadium var stora nog för att döpa den Revision 2. Vi hade vid detta tillfälle gjort stora ändringar både design- och programmeringsmässigt. Sekvenser hade lagts in för vår första funktion, ”Borrning”.

Under senare drafts arbetade vi på att göra G-Kod för Borrnings funktion. Mycket arbete lades ner för att hitta en bra lösning att hantera vektorerna i olika sekvenser.

3.2.3 Revision 3: Draft 1

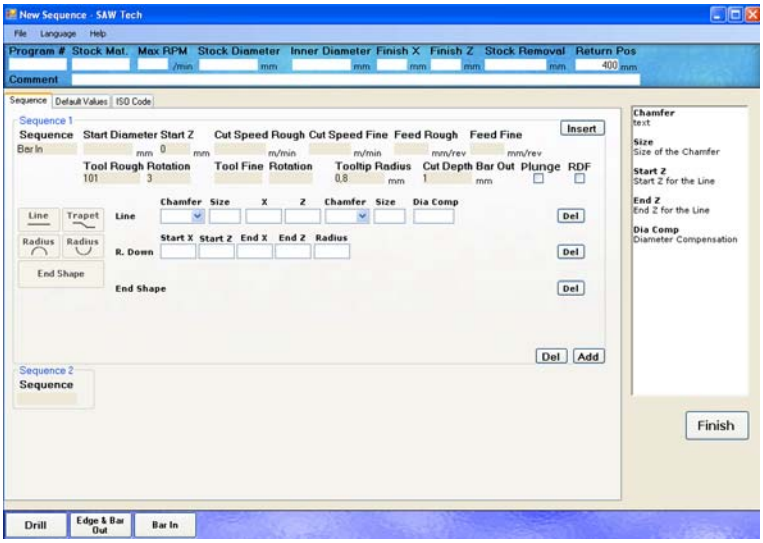
Efter att ha visat den senaste versionen av revision 2 för kunden var han generellt nöjd med hur programmet utvecklats, men han hade en del synpunkter och tillägg som han önskas läggas in.



Resultatet blev Revision 3, med fokus på layout och användarvänlighet.
En spalt för Standardvärden lades in, där användaren har möjlighet att ändra automatiska värden som används i generering av G-Kod.

3.2.3.1 Revision 3: Draft 15

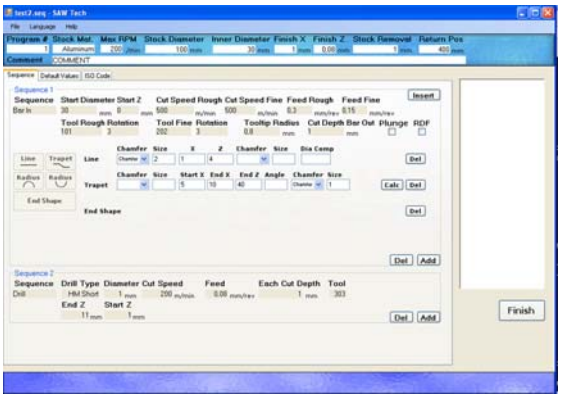
Fokus lades på att implementera nya sekvensfunktioner utöver Borrning. Först att läggas in var ”Planing och Svarvning Utvändigt”, varvid ”Svarvning Invändigt” implementerades efteråt.

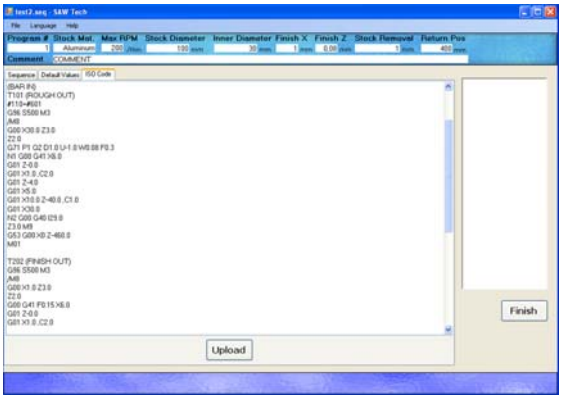


Exempel bild på hur programmet ser ut när en användare valt att använda sig av Svarvning Utvändigt funktionen.

3.3 G-Kod Generering

Anledningen till att vårt program var önskat var G-Kod generingen. Då G-Kod programmering är ingen lätt uppgift, hade vi mycket hjälp av kunden med exakta detaljer hur G-Koden skulle se ut beroende på vilka alternativ som valts i GUI:n.





Exempel på hur en sekvens som skapats med GUI:n (t.v.) ser ut, när användaren tryckt på knappen Finish, i G-Kod (t.h.).
Möjlighet att överföra denna G-kod till CNC maskinen görs genom att trycka på Upload knappen.

3.3.1 G-Kod språket

Denna sekvens generar:

1: (DRILL 2)
2: T101
3: #601=2
4: /M8
5: G97 S32000 M3
6: G0 X0 Z6.0
7: G83 Z-10 R4.0 Q4 F0.2
8: G80 M9
9: G53 X0 Z460.0
10: M01

Numreringen är enbart för att lättare förklara dess funktioner.
1: Allt i parantes är kommentarer, siffran 2 berättar diametern på borren.
2: Betyder verktygsbyte till verktyg 1 med offset 01.
3: Ger värdet av borrdiametern till en global variabel
4: ’/’ ger möjligheten att stänga/starta den efterkommande funktionen, vilket i detta fall är vatten, M8.
5: ’G97’ är initieringen av den konstanta skärhastigheten. ’S’ är parametern som bestämmer hastigheten. ’M’ bestämmer vilket håll borren ska snurra, 3 för medurs, 4 för moturs.
6: ’G0’ gör att verktyget rör sig snabbt till position X och Z.
7: ’G83’ är borrningssekvensen där den går till Z med en juckning på ’Q’ mm, varav ’F’ är dess matningshastighet.
8: ’G80’ avslutar borrningscykeln, ’M9’ stänger av vatten.
9: ’G53’ är dess koordinatsystem, där den rör sig till X och Z.
10: Stopp av sekvens.

För mer information om andra G-Koder, hänvisa till [11].

3.3.2 Java till G-Kod källkod

```
// Kollar vilken funktion som valts
if (seq[i].type.compareTo("Drill") == 0)
{
   // Om startvärdet inte har definierats
   if (seq[i].drill.txtStart.get_Text().compareTo("") == 0 ||
System.Convert.ToDouble(seq[i].drill.txtStart.get_Text()) == 0)
                        strStart =
System.Convert.ToDouble(txtSafetyApproach.get_Text()) +
System.Convert.ToDouble(txtStockRem.get_Text());
               else
                        strStart =
System.Convert.ToDouble(seq[i].drill.txtStart.get_Text()) +
System.Convert.ToDouble(txtSafetyApproach.get_Text());

      // Om Each Cut Depth inte har definierats, ta bort Q
               if (seq[i].drill.txtECD.get_Text().compareTo("") == 0 ||
System.Convert.ToDouble(seq[i].drill.txtECD.get_Text()) == 0)
                        strECD = "";
               else
                        strECD = " Q" +
System.Convert.ToDouble(seq[i].drill.txtECD.get_Text());

// Börja läsa formuläret och skriva ut motsvarande data i G-Kod
strISOCode += "(DRILL " + seq[i].drill.txtDrillDia.get_Text() + ")\r\nT" +
seq[i].drill.txtTool.get_Text() + "\r\n#601=" +
System.Convert.ToDouble(seq[i].drill.txtDrillDia.get_Text()) + "\r\n/M8
\r\nG97 S" +
System.Convert.ToInt32(((System.Convert.ToDouble(seq[i].drill.txtCutSpd.get
_Text())/System.Convert.ToDouble(seq[i].drill.txtDrillDia.get_Text())) * 320))
+ " M3 \r\nG0 X0 Z" +
(System.Convert.ToDouble(txtDrillStartClearance.get_Text()) +
System.Convert.ToDouble(txtStockRem.get_Text())) + "\r\nG83 Z-" +
System.Convert.ToDouble(seq[i].drill.txtEnd.get_Text()) + " R" + strStart +
strECD + " F" + System.Convert.ToDouble(seq[i].drill.txtFeed.get_Text()) +
"\r\nG80 M9\r\nG53 X0 Z-" +
(System.Convert.ToDouble(txtMachineZLength.get_Text()) -
System.Convert.ToDouble(txtReturnPos2.get_Text())) + "\r\nM01\r\n\r\n";
               }
}
```

3.4 Installation

Ursprungligen trodde vi att allt som behövdes var Microsofts Framework[8] för att vårt program skulle fungera, fast det visade sig att Java Redistributable Package 2.0[9] var nödvändigt. Efter att ha packat in alla nödvändiga tillägg tillsammans med vårt program, med hjälp av Installshield, var vårt program fullt funktionellt med alla Windows operativsystem.

4 Slutsatser

Efter testningar och feedback från olika användare kan vi konstatera att programmet är användarvänligt då vårt program bygger på det tankesätt som en svarvare tänker på när han jobbar vanligt. Vilket leder till att svarvaren inte ska ha några problem att anpassa sig.

Vårt program för tillfället täcker det mesta en svarvare någonsin skulle vilja framställa, och med möjligheter att lägga till specialfall för funktioner kommer detta att garantera att programmet kommer att användas.

Vi räknar med att produktionen i företaget kommer att öka då flera personer kommer att kunna ha tillgång till CNC maskinerna som använder sig av G-Kod programmering, tack vare vårt program.

4.1 Möjligheter till vidareutveckling

Programmet är fullt anpassad för att kunna vidareutvecklas med nya funktioner som skulle vilja implementeras. Möjligheten finns att HAAS, företaget som skapade CNC maskinen vi gör programmet till, blir intresserade av vårt program och en dag kanske lägger in det i deras maskiner som standard.

4.2 Kommentarer

Programmet blev större än vad vi först tänkte oss, men som i slutändan löste sig. Användningen av Microsofts Framework var ett bra val då det hjälpte mycket att arbeta med GUI:n. Det fanns stunder som vi undrade vad vi gett oss in på, men med hårt arbete och disciplin överkom vi alla hinder.

Det som vi idag tittar tillbaka till är om vi skulle valt att programmera i C# istället. Vår ursprungliga tanke var att användning av C# var ett sämre alternativ då vår kunskap inom C är mycket mindre än Java, vilket skulle leda till att programmeringen skulle gå långsammare eller i vissa fall till stopp.

5 Referenser

1) http://sv.wikipedia.org/wiki/NES [26-01-2009]

2) http://sv.wikipedia.org/wiki/CNC [26-01-2009]

3) http://en.wikipedia.org/wiki/Numerical_control [26-01-2009]

4) http://sv.wikipedia.org/wiki/CAM_(programvara) [26-01-2009]

5) http://sv.wikipedia.org/wiki/CAD [26-01-2009]

6) http://sv.wikipedia.org/wiki/J_Sharp [26-01-2009]

7) http://sv.wikipedia.org/wiki/Installationsprogram [26-01-2009]

8) http://sv.wikipedia.org/wiki/.NET_Framework [26-01-2009]

9) http://tinyurl.com/39lpq4 [26-01-2009]

10) http://en.wikipedia.org/wiki/G-code [27-01-2009]

11) http://en.wikipedia.org/wiki/G-code [25-02-2009]

6 Bilaga

6.1 Specifikation

6.1.1 SequenceCounter

```
/** Startar ett SequenceCounter objekt */
SequenceCounter();

/** Lägger till en sequence objekt i vektorn */
Void add(int X, Sequence Y);

/** Kollar om det finns ett sequence objeck på plats X */
Sequence contains(int X);

/** Tar bort en sequence objekt i vektorn på plats X */
Void delete(int X);

/** Ändrar sequence X UI:n */
Void ResizeUI(int X);

/** Gömmer meny knappar */
Void clearButtons();

/** Automatisk inskrivning av Skärhastigheter */
Void checkSeqFeedCutSpd();

/** Hantera felsöknings debug */
Void debugSeq();

/** Hantera G-Kod utskrift */
Void writeToCode();

/** Spara sekvens */
String saveAllSettings();

/** Öppna sekvens */
void readAllSettings();
```

6.1.2 Sequence

```
/** Startar ett Sequence objekt */
Sequence(int X, int Y, SequenceCounter Z);

/** Hämtar Sequence Numret */
int get_SequenceNbr();

/** Hämtar Grupp box */
GroupBox getGrpBox();
```

6.1.3 Drill

```
/** Skapar Drill objekt */
Drill(GroupBox X, Sequence Y, SequenceCounter Z);

/** Skapar Drill design */
Void createDrill();

/** Tar bort hela Drill objektet */
Void disposeDrill();
```

6.1.4 Bar In

```
/** Skapar ett Svarvning Invändigt objekt */
PSInLineTrapet(FlowLayoutPanel X, Button YDelete, Button ZAdd,
SequnceCounter S, int PNbr);

/** Skapar ett Shape Objekt Line */
createPSInLine(int X, int Y);

/** Skapar ett Shape Objekt Trapet */
createPSInTrapet(int X, int Y);


/** Skapar ett Shape Objekt Radius Upp */
createPSInRadiusUp(int X, int Y);


/** Skapar ett Shape Objekt Radius Down */
createPSInRadiusDown(int X, int Y);

/** Skapar ett Shape Objekt End Shape */
```

```
createPSInEndShape(int X, int Y);

/** Hämtar senaste Shape X */
String get_latestX();

/** Hämtar senaste Shape Z */
String get_latestZ();

/** Tar bort hela Svarvnings Invändigt objektet */
Void disposePSIn();
```

6.1.5 Edge & Bar Out

```
/** Skapar ett Planing och Svarvning Utvändigt objekt */
PSInLineTrapet(FlowLayoutPanel X, Button YDelete, Button ZAdd,
SequnceCounter S, int PNbr);

/** Skapar ett Shape Objekt Line */
createPSLine(int X, int Y);

/** Skapar ett Shape Objekt Trapet */
createPSTrapet(int X, int Y);


/** Skapar ett Shape Objekt Radius Upp */
createPSRadiusUp(int X, int Y);


/** Skapar ett Shape Objekt Radius Down */
createPSRadiusDown(int X, int Y);

/** Skapar ett Shape Objekt End Shape */
createPSEndShape(int X, int Y);

/** Hämtar senaste Shape X */
String get_latestX();

/** Hämtar senaste Shape Z */
String get_latestZ();

/** Tar bort hela Svarvnings Invändigt objektet */
Void disposePS();
```

6.2 Projektbeskrivning

6.2.1 Företaget

Legoteknik AB befinner sig i Västra Broby och arbetar med metallsvarvning. På senaste tiden har företaget inriktat sig på inköp av flera CNC, *Computer Numerical Control*, maskiner för att öka produktionen.

6.2.2 Problem

Vissa maskiner kräver avancerade programmeringskunskaper för att effektivt kunna utnyttja maskinens möjligheter. På grund av detta finns det endast enstaka personer på företaget som kan hantera dessa maskiner. Detta minskar på så sätt företagets produktion samt att maskinen inte är ständigt i drift.

6.2.3 Kundens mål

Kunden är ute efter ett enkelt användargränssnitt, som integrerat med en maskin, ska vara lätt för en vanlig svarvare att efter ett par dagars utbildning kunna utföra nödvändiga operationer.

6.2.4 Framtidsmöjligheter

I nuläget finns ingen sådant användargränssnitt på marknaden. Företagen idag använder ofta sig av leverantörens tekniker för att göra olika program på CNC maskinerna så att arbetarna kan börja producera.
På grund av detta problem, kan vår produkt vara eftertraktat av många företag samt leverantören själv.

6.2.5 Tidsplanering

Research: 220108-150208
Viktig information kring programmering för att skapa ett användargränssnitt.

Kravspecifikationer: 160208-230208
Skriva första versionen av kravspecifikationen.

Design: 240208-020308
Pappersprototyp av användargränssnittet

Programmering: 030308-300408
Programmerar enligt överenskommet användargränssnitt.

Slutrapport, Avhandling: 010508-060608