ISSN 0280–5316
ISRN LUTFD2/TFRT--5489--SE

Användarvänliga system – snabb framtagning av prototyper till grafiska användargränssnitt

Mats Carlsson

Institutionen för Reglerteknik
Lunds Tekniska Högskola
December 1993

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
December 1993

Document Number
ISRN LUTFD2/TFRT--5489--SE

Author(s)
Mats Carlsson

Supervisor
B. Nilsson LTH, L. Bosrup Alfa-Laval Thermal

Sponsoring organisation

Title and subtitle
Användarvänliga system – snabb framtagning av prototyper till grafiska användargränssnitt. (User Friendly Systems – rapid prototyping of graphical user interfaces.)

Abstract

The use of well designed graphical user interface in the world of personal computers is important today. Alfa-Laval Thermal wants their programs for their specific applications shall follow the standard to Microsoft Windows environments.

The information that is desired for the forthcoming programs can sometimes be too difficult to define and therefore hard to explicit express for the developers by potential users. With the use of early prototypes it is possible to elicit important thoughts and opinions from them. Other interesting aspects of early prototypes to evaluate are the involvement of the potential users, time, economy and the choice of programming environment.

Three cases of prototypes are here presented. They have in one way or another connection to sale support of the products of Alfa-Laval Thermal.

For the final implementation of the application it is probably best to stick to C++, the widespread standard. However, the demands on the programming language for the development of prototypes are somewhat different. There are a lot of sophisticated environments for the development of programs today, but for the specific demands of Alfa-Laval Thermal has Visual Basic from Microsoft found to be very appropriate. Visual Basic is graphically powerful, inexpensive, easy to learn, rapid changes in the prototypes are possible and Visual Basic can be used with everyone who has basic skills in structured programming.

Key words

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
Swedish

Number of pages
38

Recipient's notes

Security classification

Innehållsförteckning

1 Inledning.....1
1.1 Bakgrund.....1
1.2 Apple Macintosh.....2
1.3 Historik-grafiska användarmiljöer.....3
1.4 Microsoft Windows.....3
1.5 Problemdiskussion.....4
1.6 Avgränsningar.....5
1.7 Syfte och metod.....6
1.8 Författarens bakgrund.....6
1.9 Organisation och programvaruutveckling.....6
1.10 Hypermedia-ett sätt att presentera information.....7
1.11 Rapportens disposition.....9

2 Beaktande av standarden CUA vid design.....10
2.1 Inledning.....10
2.2 Bakgrund.....10
2.3 Den grafiska modellen.....11
2.4 Komponenter hos användargränssnittet.....11
2.5 Praktiska designprinciper.....13
2.6 Standard för utseende och beteende.....14
2.7 Dialogrutor.....16
2.8 Några övriga designbeaktanden.....17

3 Framställning av prototyper.....18
3.1 Exempel på prototyper.....18
3.2 Arbetsgång vid framställning av prototyper.....22
3.3 Arbetsinsats.....24

4 Att programmera i Windows.....26
4.1 Kort om objektorienterad programmering.....26
4.2 Programspråk som har utprövats.....26
4.3 Andra verktyg.....27
4.4 Objektorienterad programmering och rapid prototyping.....29

5 Visual Basic.....30
5.1 Allmänt.....30
5.2 Detta omfattar Visual Basic.....31
5.3 Språkliga egenskaper.....31
5.4 Fakta om DDE.....32

6 Slutsatser.....33
Litteratur

1 Inledning

1.1 Bakgrund

Mer och mer arbetsuppgifter hamnar idag på persondator. Att lära sig att använda ett mer omfattande datorprogram kan idag inte alltid vara det enklaste. I kostnaden för ett program som ska användas yrkesmässigt räknar man ofta in inte bara den direkta inköpskostnaden, utan också kostnader för utbildning och leverantörsstöd. Den tid det tar för användaren innan han eller hon kan använda programmet smidigt och effektivt och undgå att stanna upp vid smådetaljer bör också beaktas.

Många av de mer vanliga och utbredda datorprogram för persondator, som t.ex ordbehandling, kalkyl och databashantering, har funnits länge på marknaden. De utvecklades ursprunligen för datorer med 8-bitars mikroprocessorer med klockfrekvenser omkring 2-4 MegaHertz, utan hårddisk och max 64 KB internminne. Typiska sådana som förekom i sverige var t.ex Apple IIe och ABC 800 från Luxor. Utvecklarna av programvara har med tiden bättre förstått användarens situation genom omdömen av tidigare program. Samtidigt har utvecklingen på hårdvarusidan medfört att programmen blivit större och erhållit mer finnesser och funktioner så att så många användares behov som möjligt ska uppfyllas. Funktionerna har till och med blivit så många att det idag inte går att dra strikta gränser mellan olika typer program, t.ex. ordbehandlare och programvara för desk top publishing. Denna ökande komplexitet hos programmen är ett potentiellt hot mot smidigt användande av dessa.

En annan svårighet för användaren är att varje program skrivet för MS-DOS, operativsystemet för IBM PC med kompatibler, har sitt eget Graphical User Interface, GUI, och användarmetodik. Användaren av olika program måste för varje nytt, om än så väl genomtänkt, program lära sig dess specifika användargränssnitt.

Sedan lanserandet av Macintosh har Apple haft stor framgång med sin marknadsföringen av sina produkter i persondatorvärlden. Speciellt hos användare som inte vill eller kan sätta sig in i MS-DOS värld, har annamat Macintosh's användarvänliga GUI med fönster, mus, menyrad med rullgardinsmenyer etc . Alla applikationer för Macintosh följer de råd och regler som finns hos Mac Guidelines och dessa blir lättare att lära in då grundläggande kommandon är lika för alla program.

1.2 Apple Macintosh

Det grafiska skrivbordet

1984 introducerade Apple persondatorn Macintosh (dess interminne var då för denna tidpunkt ansenliga 128 KB). Denna dator var den första som i större omfattning spred idéerna om en intuitiv, grafiskt orienterad användarmiljö hos datorer. Idén är att bildskärmen ska representera ett skrivbord, där det finns olika dokument. Dessa dokument kan flyttas, ändra storlek och minimeras till s.k ikoner av användaren. Istället för att användaren ska lära in ett antal tangentkommandon i ett textbaserat system, finns här en s.k mus med vars förflyttning på en bordsskiva man styr den s.k muspekaren på bildskärmen. Muspekaren representerar så att säga användarens hand på skrivbordet. Med musen utför användaren olika grafiska kommandon till programmet. Kommandon, speciellt de som användes sällan, blir lättare att minnas. Grafiska kommandon behöver till skillnad mot textbaserade kommandon inte memoreras i minsta detalj. Det räcker ofta att användaren kommer ihåg huvuddragen i kommandot. Ett genomtänkt användargränssnitt av denna typ ska tillåta - och stimulera - utforskande utan att operationer med oåterkalliga resultat utförs av misstag.

DOS-motstånd

Många rutinerade och inbitna DOS användare har haft svårt att fördra Apple Macintosh. En del har kallat den för spel och lek dator, dvs de tycker det är tidsmässigt ineffektivt att låta långsammare grafiska kommandon ersätta korta (och mer eller mindre kryptiska) textkommandon de redan behärskar. En farbror till mig tyckte också en gång att : " Macintosh får sekreterare tro att dom behärskar datorer ". Med det avsåg han förmodligen något med typ av att användare av Macintosh har tagit en alltför lätt genväg till datoranvändande medan han själv minsann med stor möda lärt sig de vanligaste DOS kommandona och att han vid behov måste haft tålamod att dyka djupt ner i manualerna. Min släkting verkar dock ha ändrat åsikt sedan han installerat det Macintosh-liknande Windows och sedan börjat att utforska kalkylprogrammet Excel. Tiden verkar mot de konservativa DOS-entusiasterna. Allt mindre ny programvara görs för DOS.

1.3 Historik-grafiska användarmiljöer

Idéerna bakom Macintosh var inte helt nya. Tidigare hade Apple utvecklat persondatorn Lisa som var en föregångare till Macintosh, men denna misslyckade kommersiellt. Konceptet med fönster, mus etc, kommer ursprungligen från utvecklandet av användarmiljön hos programmeringsmiljön i språket smalltalk som utvecklades av Xerox Park under 70-talet. Apple köpte därefter rättigheter för att kunna utnyttja dessa .

1.4 Microsoft Windows

Microsoft Windows, hädanefter i rapporten benämnt som Windows, är ett sådant Macintoshliknande användargränssnitt avsett att användas tillsammans med DOS. Det har funnits sedan 1985, men det stora genombrottet kom först med version 3.0 som lanserades våren 1990. Den senaste versionen 3.1 innebär förbättringar för först och främst stabiliteten. I version 3.0 kunde det ibland hända att hela systemet låses, vilket medförde att användare ofta fick lagra den senaste versionen av det pågående arbetet. Dock gäller fortfarande att version 3.1 ej är ett äkta multikörningssystem och är då fortfarande inte är helt tillförlitligt. Version 3.1 innebär också ökad snabbhet och förbättrad smidighet i användandet, då framför allt av den s.k filhanteraren, där man förbättrat en del funktioner. Det är dock version 3.0 som banat väg när det gäller att tillföra "fönster & mus"-teknik till det stora antalet användare av IBM PC och dess kompatibler.

Då Windows är som ett skal som ligger ovanpå DOS, innebär det sämre prestanda och att Windows begränsas av ofullständigheter hos det mer än ett decennium gamla DOS. Det är först med 90-talets typiska standard för persondatorer som Windows har kunnat fungera någorlunda smidigt, dvs minst 1 MB minne och 286-processor. Även med denna hårdvarukonfiguration kan Windows upplevas som långsamt och mer minne och minst 386-processor är i praktiken ett måste. Dock bör man nämna att systemmjukvaran hos Macintosh också lär ha "skönhetsfläckar", vilket t.ex medfört problem med att använda äldre program då en ny version av operativsystemet kom. Även här har alltså påbyggnader för möta nya krav medfört en snårig struktur.

Vid en jämförelse med Macintosh ger användandet av Windows ett mindre intuitivt intryck av grafiska handlingar på ett skrivbord. Detta beror till en del på att Microsoft ej vill att Windows ska likna Macintosh alltför mycket på grund av upphovsrättsliga hot från Apple, men också på det underliggande DOS. För en van användare av Macintosh utan erfarenhet av DOS kan t.ex filhanteringen vara

ganska jobbig. Andra skillnader beror på att Windows följer IBM´s s.k CUA-standard. Denna ger regler och råd för "fönsterstruktur", men också normer för tangentbordskommandon. IBM´s CUA-standard för tangentbordets funktion i samspel med fönstermiljö är omfattande. Detta innebär att man kan alltid arbeta med mus, men för många Windowsprogram går det också bra utan. I många fall blir motsvarande tangentbordskommandon otympliga och svåra att komma ihåg, men ibland kan Windows för en del standardkommandon fungera snabbare och smidigare än med mus. Exempel på detta är ordbehandlaren Word och kalkylprogrammet Excel. De finns både för Macintosh och Windows. I Windows kan man använda dessa program utan mus. Många standardoperationer blir dock mycket komplexa och att ha manualerna i närheten är ett måste, men detfungerar om man ej vill eller kan använda musen. I Macintosh-versionerna är möjligheterna med tangentkommandon betydligt mindre omfattande. För mer grafiskt inriktade applikationer är musstöd även i Windows ett måste.

1.5 Problemdiskussion

Program för speciella tillämpningar som t.ex Alfa-Laval Thermals program som användes av försäljare ute på fältet ökar problemen för utvecklandet av programvara då det inte alltid finns något befintlig programvara, vars struktur man göra om eller bygga vidare på. Om användaren kan tycka att manualerna av vanliga standardprogram ibland kan vara svårförstådda, kan de när det gäller sådana här specialgjorda program upplevas som obegripliga. Problemet ligger här i specifikationen. Manualen kan vara något som man sätter ihop i efterhand på för kort tid när allt annat arbete är utfört av de som kan produkten utan och innan. Det är här lätt att dra paralleller till instruktionsböckerna avsedda för videobandspelare, i synnerhet instruktionerna för timerinspelning ...
Användarna kan på grund av detta inta den motsatta åsikten att användarhandledningen borde ha varit klar innan något annat arbete påbörjats.

Problem med design av användarvänliga program gäller ej enbart GUI. Det kan vara svårt att locka fram ur användaren vad denne egentligen vill ha i ett kommande program, för de tänker ej i strukturerad programkod. Och även om användaren kan precisera vad denne vill för ögonblicket, så kan det vara svårt att veta hur arbetet som programmet ska utföra kan komma se ut i framtiden, både med avseende på förändrade insikter om hur den aktuella informationen ska behandlas och framtida datorteknik.

För utveckling av större informationssystem i stor- och minidatorvärlden kan det krävas erfarna konsulter med kunskaper i management och organisation, som t.ex Andersen & Co eller McKinsey, som kan strukturera informationen och dess användning innan man påbörjar fasen med implementering av programkod. För per-

sondatorer är systemen mindre komplexa, de har kortare livslängd och därmed mindre krav på underhåll, etc. På denna nivå kan man mer tänka sig ett mer intuitivt WYSIWIG-koncept (what you see is what you get) där användare deltager interaktivt i utvecklingsprocessen.

Detta examensarbete handlar om detta, dock begränsar det sig till att göra olika utkast på GUI´s som sedan ska tjäna som utgångspunkt för diskussioner om framtida program. Då datorstandarden för de potentiella programmen baserar sig på operativsystemet MS-DOS, har MS-Windows använts.

1.6 Avgränsningar

Användarvänliga system är för detta arbete avgränsat till begrepp som:

- **Tidiga prototyper**. Bl.a vill Alfa-Laval Thermal med detta arbete att man för kommande program bättre ska kunna "låsa" specifikationerna innan man låter utomstående programmerare ta över arbetet. Vid tidigare projekt har man ej tillfredställande fått fram tillräcklig information inom rimlig tid innan implementering av programkod påbörjades. Man vill få en inblick hur de slutgiltiga applikationerna kan se ut på ett tidigt stadium. Genom bättre specifikationer hoppas man få mindre efterkloka omdömen.

- **Enhetlighet**. Beteendet hos applikationen bör följa CUA-standard vilket bl.a underlättar inlärning och användning.

- **Enkel, grafisk, intuitiv, snabb, förlåtande...**
  Detta är nyckelbegrepp som karaktäriserar det bästa hos Macintosh. Nu när möjligheterna finns för PC-världen, vill Alfa-Laval Thermal inte vara sena med att hoppa på Windows-tåget. Detta kan innebära vissa konkurrensfördelar och ökad trivsel hos personalen.

- **Bärbara datorer**. Prototyperna syftar till att utveckla system som ska användas av t.ex försäljare och servicepersonal.

1.7 Syfte och metod

Frågeställningar som önskas utredas är :

- Hur är det att arbeta med utvecklingsverktyg, och vilka krav på dessa som ställs för prototyper respektive färdigt program.
- Hur det fungerar med att arbeta med prototyper.
- Hur arbetet med prototyper kräver användarengagemang, tid, ekonomi etc.

Metoden, med vilken detta arbete genomfördes, var att praktiskt försöka åstadkomma 2-3 tidiga prototyper, både sådana som hade en utgångspunkt i delar av befintlig programvara unik för Alfa-Laval Thermal men också sådana som mer var visioner och kunde baserade sig på lösa skisser på papper.

1.8 Författarens bakgrund

Jag hade innan detta arbete aldrig använt Windows och knappt ens använt PC, och denna användning av PC innebar ej någon större kontakt med DOS. Att förstå de grundläggande kommandona i DOS var dock inget problem eftersom jag tidigare använt mig av textkommandon i Unix på arbetsstationer och i VMS på minidatorterminaler(VAX). Jag har följt kurser i objektorienterad programmering och i realtidsprogrammering. Den förra har gett förståelse för hur objektorienterade språk används i Windows, medan den senare har gett insikter om principer för s.k multikörningssystem som Windows ska efterlikna, det är ju inte ett äkta sådant. Vana vid mus- och fönsteranvändande har kommit från X-Windows hos arbetsstationer och naturligtvis från Macintosh. I den objektorienterande kursen gjordes också ett projekt i programmering för Macintosh. Fokus låg dock inte där på programkod för hanterandet av det grundläggande gränssnittet, även om inblickar i detta gavs.

1.9 Organisation och programvaruutveckling

Utveckling av programvara, och då speciellt för affärsändamål, berör människor på olika avdelningar och positioner hos Alfa Laval Thermal i Lund och det kan därför vara intressant att försöka få en inblick i strukturen hos organisationen i syfte att inringa faktorer som påverkar utvecklandet av programvara.

Personaldirektören hos Thermal kunde vid en förfrågan endast ge en grov och översiktlig plan över organisationen och detaljerade beskrivningar för olika befattningar fanns överhuvudtaget ej. Organisationen är indelad i divisioner för olika segment av marknaden för värmeväxlare. Vissa funktioner t.ex personal och ekonomi är centrala och delas av divisionerna. Vid sidan av denna hierarkiska organisaton kan människor bindas samman i olika projekt, vilket innebär att personal från olika avdelningar, och även olika divisioner, med varierande position arbetar tillsammans. Personaldirektören ville därför kalla detta en " nätverksorganisation ". Informella vägar för kommunikaton är alltså här viktiga och det kan ta lång tid för utomstående att verkligen förstå hur saker och ting verkligen fungerar rent organisatorisk. Det är här viktigt att veta vem som känner vem. Detta gäller i högsta grad för utomstående konsulter. Enligt personaldirektören används de delvis som "bollplank". Detta kan innebära att konsulterna kommer med synpunkter på deras egna ideér eller ger uppslag på ideér som de sedan själva prövar och utvecklar.

Inom t.ex Systemvetarlinjen, som är en datautbildning inom sektorn för den ekonomisk-administrativa utbildningen, lär man sig hur man utreder en organisation för att förstå dess behov av datoranvändning. Konsulter inom ADB med detta perspektiv kan därför här stöta på svårigheter och ha svårt för att inom rimlig tid komma till några slutsatser.

Det kan också allmänt sägas att människor med teknisk bakgrund i det tekniskt inriktade Alfa-Laval Thermal kan ha svårt att kommunicera med ADB-människor som har en annan bakgrund. Det är alltså föga förvånande att människor hos Alfa Laval Thermal med inflytande över datafrågor ofta har tekniker bakgrund.

Detta arbete handlar kanske inte i så hög grad om teknisk programmering utan kunde kanske i princip ha utförts av någon med ADB-bakgrund. Det har emellertid här nog stor betydelse att kunna samtala på "teknikers vis", inte minst när det gäller den tekniska information som ska presenteras i programmen avsedda för olika former av kundkontakter

1.10 Hypermedia -ett sätt att presentera information

Detta arbete tar ju upp frågor som berör presentation av information och det kan därför vara intressant att här presentera begreppet hypermedia, vilket vi mer och mer kommer att komma i kontakt med i framtiden

Hypermedia är ett sätt att binda samman information genom att använda associativa länkar. Ett hypermedia system kan liknas vid en databas med ett grafiskt användargränssnitt. Men istället för de typiska fält- och filstrukturerna, är infor-

mationen fritt strukturerad och sammanbunden av ett nätverk av associativa länkar. Datainnehållet är ej heller begränsat till text och siffror, utan det kan också vara grafik , fotografier, ljud eller video. Andra begrepp som nämns i detta sammanhang är hypertext och multimedia och dessa avser inte riktigt samma sak.

Hypertext är hypermedia avgränsat till textinformation. Hjälpfunktionen hos en del större applikationer för Windows kan ses som exempel på hur hypertext kan fungera. Man kan här leta sig fram till den önskade informationsavsnittet genom att man går neråt i ett hierarkiskt grenverk på samma sätt som man blickar över en boks innehållsförteckning översiktligt för att sedan närmare granska underrubriker hos ett kapitel som kan omfatta den sökta informationen. Den sökta informationen finns då i en nod i grenverket. Detta grenverk utvidgas till ett nätverk genom att en nod kan ha associativa länkar eller referenser till andra noder i horisontell eller vertikal led. I t.ex Word ligger dessa referenser nära, men strukturen av hypertext blir mer påtaglig om dessa associativa länkar hänvisar till mer avlägsna delar av informationssystemet. Detta märks mer i hjälpfunktionerna för programmeringspråk i Windowsmiljö. Hänvisningar till centrala begrepp och definitioner är återkommande i många noder. Genom att slumpvis följa de associativa länkarna kan man här följa en bana som i motsvarande bok går kors och tvärs över kapitelindelnigen.

Multimedia i sin tur behöver inte innebära att associativa länkar existerar utan detta är mer ett begrepp för att presentera information som samtidigt är i form av text, bilder, ljud och/eller video.

Den "Tutorial", den på mjukvara baserade "lär känna ditt program" översikt, som medföljer många större applikationer för Windows, är ett annat praktiskt exempel på begreppet hypermedia. Här kan man för t.ex kalkylprogrammet Excel få en översikt vad man kan uträtta med detta program demonstrerat med exempel som ibland kräver användarens medverkan. Man kan här själv välja vilket arbetsområde som kan vara intressant att få en översikt av och hur djupt man vill fördjupa sig i detta.

Hypermedia har visat sig vara ett idealt sätt att tillgå stora mängder av information och dess utvecklig har gått hand i hand med Compact Disc teknologi för att åstadkomma billiga och lättanvända informations och dokumentationssystem.

Idag finns det prisvärd hårdvara att tillgå, för hushållen har Philips nyligen lanserat sitt CD-I system och för persondatorer finns det också på CD baserade multimediasystem. Det intressanta är dock när sådana system är allmänt förekommande för bärbara persondatorer, bärbara CD-spelare är nämligen ganska batterikrävande

1.11 Rapportens disposition

Nästa avsnitt tar upp CUA-standard i detalj, men det går dock bra för den som förstår grunderna i Windows att förstå resten av rapporten utan att ha läst detta.

Avsnitt 3 behandlar prototyperna, det direkt praktiska resultatet av detta arbete. Dessa är användargränsnitt till program för dimensionering av kylcykel, för konfiguering/dimensionering av värmeväxlare och för en databas för kundorder.

För den som är är intresserad av programmeringsverktyg ges en översikt i avsnitt 4 där bl.a objektorientering ,Turbo Pascal, C++ tas upp och avsnitt 5 ges en något djupare inblick i Visual Basic.

Till sist presenteras några slutsatser i avsnitt 6

2 Beaktande av standarden CUA vid design

2.1 Inledning

Ett av de grundläggande skälen för att utveckla Windows-baserad programvara är dess smidiga inlärning och användning. Användare av Windows upplever en signifikant produktivitetsökning huvudsakligen på grund av att funktioner gemensamma för många program utförs på samma sätt. T.ex. Edit-menyn har samma poster och samma tangentkommandon för snabbval för en viss menypost i ordbehandlare,färgläggnings- och ritprogram.Lite tillspetsat uttryckt, om du vet hur man använder ett Windows-program, vet du hur man använder alla.

För en användare räcker det med att denne är bekant med huvudragen av utförande och beteende i Windows för att hanterandet av detta ska fungera smidigt. Tillvaron är mer komplex för den som ska åstadkomma program med CUA standard som ledstjärna. Det finns en mängd definitioner och begrepp som ej bör glömmas. Detta avsnitt tar upp viktiga punkter vid design av Windows enligt CUA.

2.2 Bakgrund

För formalisering av riktlinjerna för allmäna funktioners utförande, har IBM publiserat en uppsättning regler och råd som är avsedda att användas vid design av användargränssnitt, kallad Common User Access, CUA. CUA är an av delarna i System Application Architecture, SAA från IBM. CUA är tänkt för program skrivna för OS/2 Presentation Manager, men CUA's riktlinjer är lika ändamålsenliga för program skrivna för Windows. IBM's avsikt är att SAA ska tillåta kunder att använda sina program i olika datormiljöer (hårdvara, operativsystem etc). CUA är en grundstomme för konsekvent tänkande över program som kan resultera i minskad tid för inlärning, förbättrad produktivitet och ökad tillfredsställelse hos användaren.

2.3 Den grafiska modellen

Alla program har fyra aspekter som beskriver deras gränssnitt för användare:

- **Presentation**: vad användare ser.
- **Interaktion**: hur användare interagerar med komponenterna.
- **Funktioner**: hur liknande funktioner implementeras på samma sätt.
- **Processekvens**: hur användare och datorn kommunicerar med varandra.

I den grafiska modellen är processekvensen objektfunktions orienterad. Det innebär att användaren väljer ett objekt och sedan väljer en funktion som lämplig till detta objekt. När man konsekvent stödjer en objektfunktions process sekvens, förstärker man användarens begreppsmässiga modell av användarens gränsnitt.

Objektfunktionens processekvens har fördelar över funktionsobjekt sekvensen, där användaren först väljer en funktion och sedan den information som denna ska arbeta med. Några av fördelarna hos objektfunktionens processekvens är:

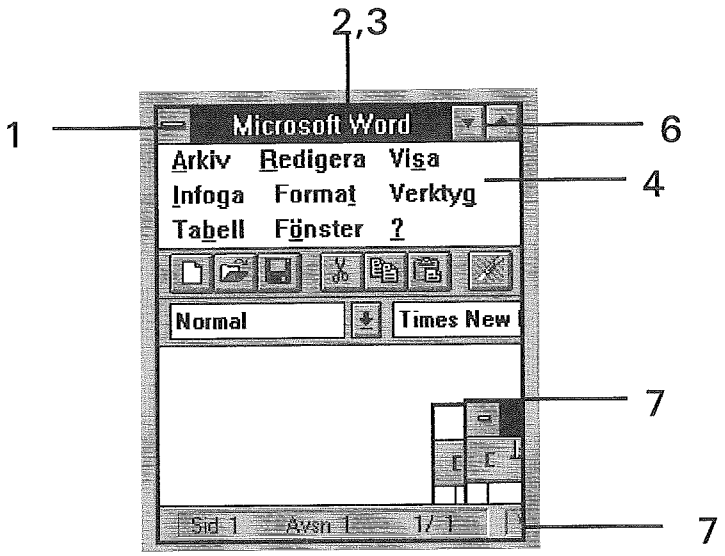
- Den förser användaren en grund att utforska programmet genom funktioner som är sammankopplade för ett visst sammanhang. Användaren kan först välja ett objekt och sedan skumma igenom funktionerna hos programmet för att se vilka som hör till det valda objektet.
- Den tillåter användaren att utföra en serie av handlingar hos ett utvalt objekt snarare än att repetetivt välja objektet för varje önskad handling. T.ex hos en ordbehandlare brukar användare vanligtvis välja en del av en text (ett objekt) och sedan anbringa det till några stilar (funktioner) som fetstil, kursiv och understruket. Användare finner ett gränssnitt obekvämt och repetetivt om de måste välja ett objekt många gånger för att begära alla handlingar de önskar utföra på ett objekt.

2.4 Komponenter hos användargränsnittet

Detta är bekant för alla Windows-användare även om säkert många ej utryckligen känner till definitionerna och funktionerna. Dock kan det t.ex vara intressant för Macintosh-användare och de som någon gång sett en demonstration av ett Windows-program att ta upp detta.

Fönstrets delar

Alla applikationer och vissa dokument som man väljer att arbeta med öppnas som separata fönster. Alla fönster har vissa gemensamma element. Alla fönster använder dock inte samtliga element.



1. *Systemrutan* är placerad högst upp till vänster i fönstret.Med kommandona i *systemmenyn* kan man växla till ett annat program, stänga fönstret, samt ändra läge och storlek på fönstret utan att använda mus.

2. *Titelraden* visar namnet på applikationen eller dokumentet.Fönstret flyttas då man med musen drar i denna.

3. *Fönstertiteln* kan vara namnet på en applikation, en grupp, en titel, en katalog eller en datafil.

4. *Menyraden* visar tillgängliga menyer.

5. *Bläddерkanterna* används för att flytta fram delar av ett dokument när inte hela dokumentet syns i fönstret .

6. *Maxi-* och *Miniknapparna* maximerar det aktiva applikationsfönstret så att hela "skrivbordet" täcks eller minimerar fönstret till en grafisk symbol.

7. Med *Fönsterkanter* och *Fönsterhörn* kan man med dragrörelser hos musen ändra fönstrets storlek.

Kontroll element

En kontroll (svengelska!) är ett gränssnittselement som tillåter användare att välja alternativ och skriva infomation. Kontroller ger konsistens i både presentation och interaktion i likhet med andra gränsnittselement. Varje kontroll har ett unikt utseende och ger användaren ett specifikt sätt att interagera med den.

Kontroller används först och främst i dialogrutor för att be användaren om ytterligare information för ett begärt kommando. Dessa kontroller kan naturligtvis också användas i ett fönster.
De vanligaste kontrollerna är:

- Kryssruta (Check box)
- Alternativring (Radio button)
- Tryckknapp (Push button)
- Listruta (List box)
- Kombinationsruta (Combo box)
- Textruta (Text box)

2.5 Praktiska designprinciper

Det finns två primära mål med applikationer som följer CUA, se till att göra användare förtrogna med resultaten av deras handlingar och att låta dem kontrollera händelseflödet.

Förse användaren med väntade reaktioner

Det bästa sättet att få användaren att känna förtrogenhet är att försäkra sig att applikationen reagerar på ett konsekvent, pålitligt sätt.

Förslag till detta inkluderar:

- **Använd metaforer**. Om man modellerar programmets sätt att te sig med någonting som användaren är van vid, sådant som ett existerande system baserat på papper, kommer de att få en bättre förståelse av programmets begreppsmässiga modell. Man bör alltså inte i onödan ändra i arbetsgången för t.ex ett väl fungerande blankettifyllningsystem. Dessutom kommer användaren att snabbare att känna trivsel med applikationen.

- **Var konsekvent**. Använd kända kommandon (välja menyer,klicka ikoner osv.) för välbekanta funktioner.Se till att skärmbilder för närliggande uppgifter liknar varandra.

- **Undvik "modes"**. Ett "mode" är ett tillstånd där användaren har begränsad tillgång till vissa delar. Det används ofta felaktigt för att tvinga användaren i en speciell riktning. Det kan vara bra med modes för korta felmedelanden eller under "verktygs"-val. T.ex då muspekaren i ett ritprogram är en penna kan användaren rita men ej dra omkring figurer. Förse visuell bekräftelse på moden, som t.ex en annan muspekare eller gråa menyval.

- **Tillåt användaren att kontrollera flödet.** Användaren kommer att känna sig mer förtrogen med och i kontroll av applikationen om de kan bestämma sin arbetsväg. Att tillåta användaren att känna sig ha kontroll tar bort en del av mystiken kring datorer.

- **Lista alternativ visuellt.** Förse användaren med en visuell lista av tillgängliga funktioner som användaren kan uttnyttja. Bekanta metoder inkluderar an palett av verktyg eller färger och en meny av alternativ.

- **Tillgodose återkoppling hos användaren.** Varje användarhandling bör resultera i en visuell eller akustisk respons.

- **Uppmuntra utforskande.** Straffa ej användare om de vill utforska alternativ. Ofta är det det bästa sättet att lära sig ett nytt program. Tänk dock på att inkludera ett *Ångra* alternativ.

2.6 Standard hos utseende och beteende

Mycket av Windows standard för applikationers utseende och beteende är tydligt i Progamhanteraren och de medföljande programmen (Paintbrush, Control Panel osv). T.ex så framträder varje applikation i ett huvudfönster. Enkel- resp dubbelklick med mus har i allmänhet samma effekt.

- **använd objektfunktions processen** som nämnts tidigare.

- **Välj fönster med omdöme.** Huvudfönstret skall binda sammanbinda applikationen och vara dess ryggrad. Sekundära fönster, som pop-up fönster och dialogrutor,bör frambringas av huvudfönstret som ett resultat av någon handling i huvudfönstret. Pop-up fönster kan reprensentera en helt separat modul av applikationen, men de bör till skillnad mot huvudfönstret ej kunna minimeras till grafiska symboler.

- **Visa val av objekt.** När ett objekt har blivit valt med ett musklick,förse användaren då med en visuell ledtråd, som t.ex rektanglar på dess hörn eller en förändring i färg eller tjocklek. Kännetecknet för urval skall inte märka ut objektet permanent. Markeringen för urval ska behållas när användaren utför en handling på objektet. Detta tillåter användare att utföra fler handlingar. Markeringen ska tagas bort när användaren väljer ut någonting annat.

- **Begränsa inte muspekaren.** Muspekaren bör vara fri att röra sig från ett fönster eller program till ett annat. Dess rörelse bör ej begränsas. Dess rörelser får ej vara oberoende av musen. Detta bryter mot den fysiska relationen mellan mus och muspekare.

- **Använd tangentbordets markör .** Denna markör märker ut den punkt kommande nedslag från tangentbordet kan komma att påverka. T.ex så reprensenterar den blinkande markören i en ordbehandlare tangentbordets markör. Tangentbordets markör existerar oberoende av muspekaren. De förenas emellertid vid musklick.

- **Markera focus för inmatning.** Focuset för inmatning är den punkt på skärmen som för tillfället tar emot inmatning från tangentbord eller skärm. T.ex, om användaren väljer en rullgardinsmeny, så har den menyn focus. Om användaren markerar text, så har den texten focus. Objektet som har focus skall alltid markeras. T.ex. brukar valda menyposter vanligtvis markeras med inverterad video (vit text mot svart botten) och kontroller i en dialogruta blir omgärdade med en prickad linje.

- **Skapa ett gränssnitt för tangentbordet.** Piltangenter, tabbar, och tangentkombinationer ska vara ett alternativ för användare som vill utföra mindre antal muskommandon. T.ex så har varje menyval (och ibland trycknappar) en bokstav understruken. Detta innebär att tangentkombinationen ALT + den understrukna bokstaven för en menypost innebär samma sak som att utföra menyvalet. Till menyposterna går det också lägga till ytterligare en tangentkombination innehållande antingen en enda funktionstangent eller *ctrl/shift/alt* + ytterligare tangenter.

- **Följ konventioner för musklick.** Windows använder, men påtvingar ej, vissa konventioner för musklick. I allmänhet användes den vänstra musknappen för val och behandling av objekt och den högra knappen kan ibland användas för andra ändamål som att växla mellan tillstånd eller att frambringa ett pop-up lista av alternativ. För den vänstra musknappen gäller följande konventioner:

Klick väljer ett objekt.

Dubbelklick väljer ett objekt och medför någon standardmetod eller underförstådd handling.

Klick-drag (knapp ner + dragrörelse) förändrar t.ex. objektets storlek och position eller väljer och framhäver text.

Ctrl-Klick tillför det valda objektet till en mängd av tidigare valda objekt.

Shift-Klick utökar en rad av tidigare mängd valda objekt mellan mängdens första objekt (dess ankare) och objektet som klickats(Se t.ex Filhanteraren i Windows t.ex.)

- **Menyer.** Windows är redan försedd med och framhäver en ganska sträng standard för framträdande hos rullgardinmenyer. Det finns emellertid en del utrymme för stilar man kan välja och de alternativ man vill införa. T.ex så ska man använda unika mnemonics av tidigare nämnd Alt+bokstav typ för varje topp-nivå och rullgardinernas menyposter. Ett *Transumtecken* ... följer en menypost vars val resulterar i en dialogruta. *Skuggat* kommandonamn användes som bekant för icke tillgängliga kommandon.

De flesta applikationer, antingen de är för grafiska ändamål, undervisning, ekonomi, vetenskap eller tekniska ändamål, utför en mängd typiska operationer för filhantering, editering och tillträde till hjälpsystemet. På grund av detta kräver CUA särskild uppläggning av *Arkiv, Redigera,* och *Hjälp* menyerna. I tillägg till detta erbjuder den vägledning för uppläggning av *Granska* och *Alternativ* menyerna.

2.7 Dialogrutor

Det finns så många sätt konstruera dialogrutor så det finns få generella råd. Det är emellertid en bra ide att välja en viss stil och att hålla sig till denna i en applikation.

Det finns två typer av dialogrutor, modala och mode-lösa.
Modala, som stänger av funktionen av applikationen tills operationen i dialogrutan slutförts, bör användas i flertalet fall. Mode-lösa, vilka tillåter att programmet kan fortsätta när dialogrutan syns, bör användas för speciella ändamål, t.ex för att kontrollera rättstavningen i ett dokument i en ordbehandlarare. CUA kräver dock en speciell typ av dialogruta för *Öppna* och *Spara* som dialogrutor.

2.8 Några övriga designbeaktanden

- **Välj storlek för huvudfönstret.** Bara den yta som krävs för att presentera den önskade informationen ska utnyttjas samtidigt som den ska gå att använda den smidigt,

- **Använd rätt metod för omritning.** Det finns två metoder för att rita om ett fönster när dess storlek ändrats. Innehållet i fönstret kan behållas i dess originalproportioner. Delar av innehållet kan då låtas bli "klippt" om fönstret blir för litet. Innehållet kan också formas om så att det alltid passar bra inom fönsterkanterna. Detta alternativ passar bra för t.ex spel.

- **Utnyttja färger måttligt.** Färger bör förstärka en etablerad kommunikation med användaren. De bör användas konsekvent och användaren kan kanske få möjligheten att förändra dem. Man bör tänka på hur färgerna ter sig i en skala med gråtoner. Skärmarna på bärbara persondatorer kommer ju de närmaste åren fortfarande starkt domineras av monokroma LCD-paneler.

3 Framställning av prototyper

3.1 Exempel på prototyper

Tre stycken prototyper med anknytning till Alfa-Lavals verksamhet kommer här att presenteras. Prototyperna här, utom den första, har gjorts med Microsoft´s Visual Basic. Detta är kanske inte det mest optimala verktyget för att göra fullständiga prototyper, men med detta är det enkelt att åstadkomma utkast till användargränssnitt och man bör dessutom inte binda sig vid ett speciellt utvecklingsverktyg i det snabbt växande utbudet av programmeringsverktyg för Windows.

Kylcykel

Bakgrunden till denna prototyp är ett program skrivet för DOS i Quickbasic. I detta program dimensionerar man en kylanläggning. Det som här är av intresse för Alfa-Laval Thermal är att i detta system ingår värmeväxlare, som är

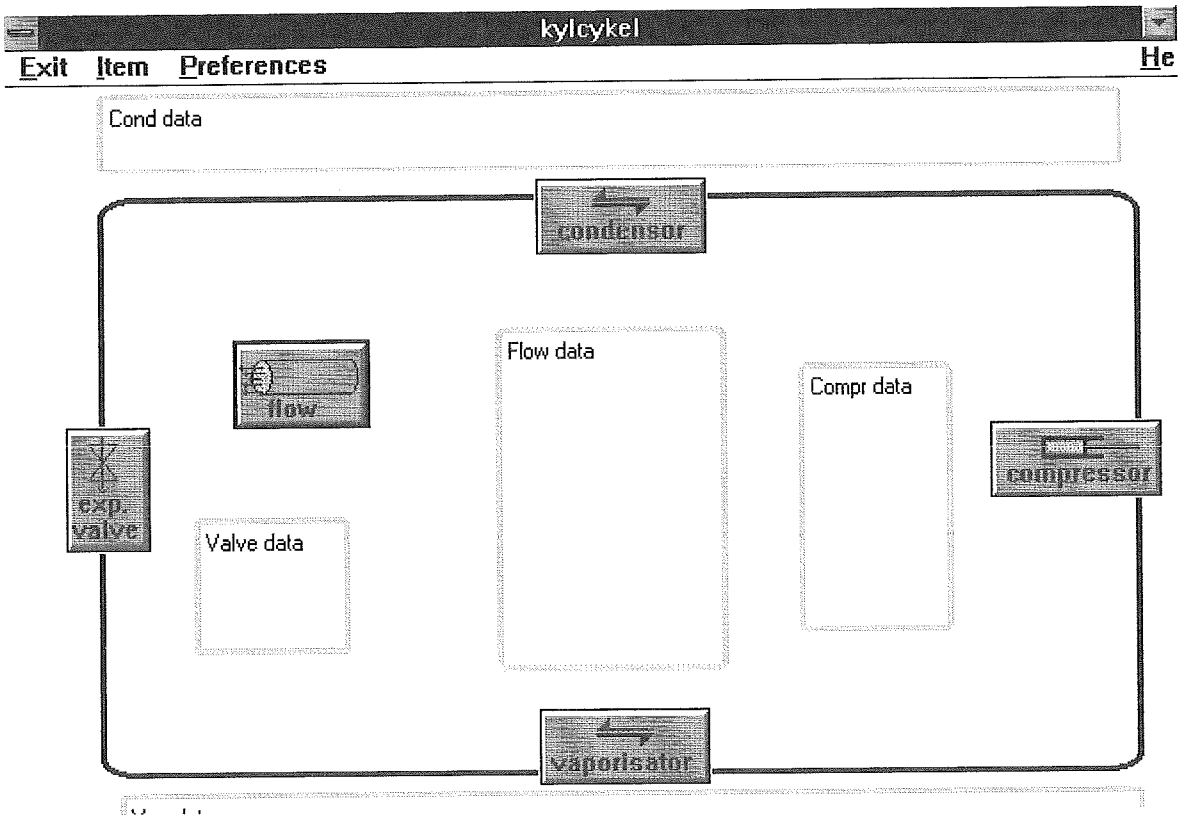


fig. 1 kylcykel

Thermal´s produkt. Med denna typ av program vill man underlätta försäljningsarbetet med kunder som vill köpa värmeväxlare för användning i kylanläggningar. I programmet anger man t.ex först önskad kyleffekt och önskade in- och utloppstemperaturer för det medium som ska kylas. Därefter provar man iterativt att ange andra data i kretsen, så att programmet kan beräkna alla viktiga data som berör värmeväxlarna. Denna prototyp var den första som gjordes och den skrevs i Turbo Pascal för Windows. Indata ges till programmet genom att man trycker på en av trycknappar med grafik som reprensenterar en del av kretsen och då poppar ett fönster upp där man kan skriva in eller avläsa data som rör denna del. Utseendet hos denna prototyp gillades då det var betydligt mindre plottrigt än hos ursprunget samtidigt som det anslöt till grundläggande CUA-struktur i både utseende och ytligt arbetssätt.

Det bör dock påpekas att denna prototyp var ett utfoskande i möjligheterna med Turbo Pascal. Detta innebar att mycket tid ägnades åt att försöka förstå hur man kunde överföra olika ideér till detta programspråk. Det tog lång tid för mig att med denna version förstå hur jag skulle implementera vissa detaljer. Följden blev att ideér om hur man ska arbeta med gränssnittet glömdes bort, vilket ingen på Thermal ej heller tänkte på utan. Denna punkt togs dock upp av min handledare (hos reglerteknik) som har erfarenhet att värdera arbetsmetodiken hos betydligt mer komplexa och sofistikerade grafiska användargränssnitt.

Specifikation av plattvärmeväxlare (PHE)

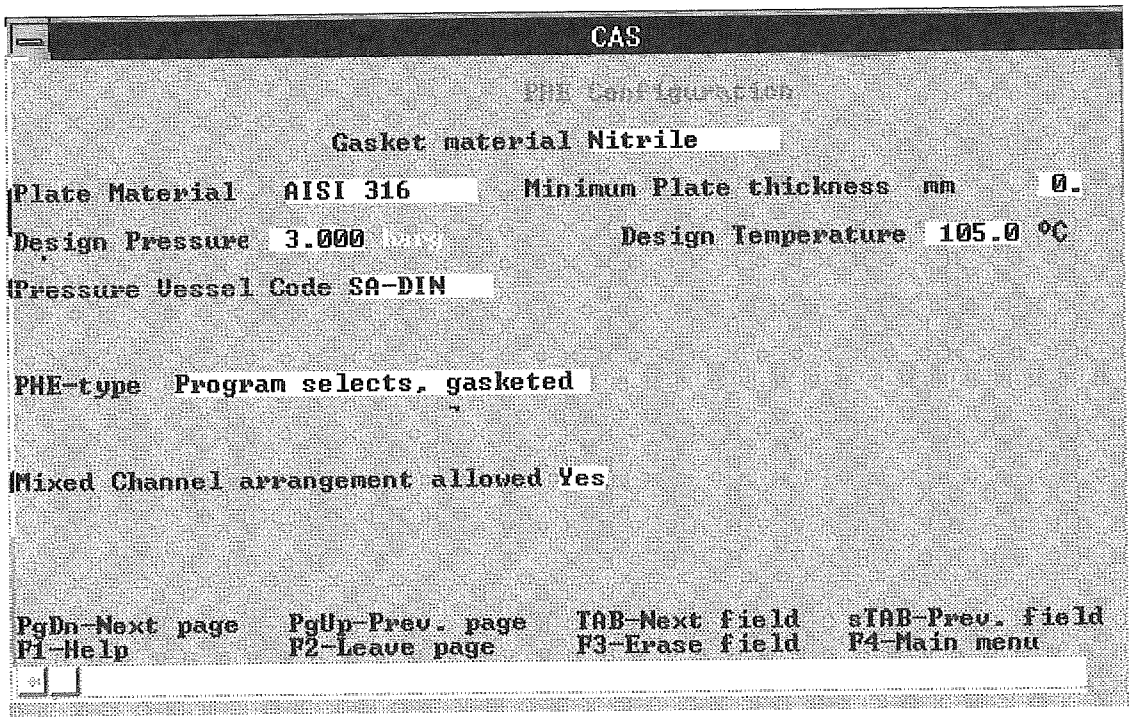


fig. 2 CAS

Bakgrunden till denna prototyp är ett avsnitt ur CAS, ett program avsett för säljstöd av plattvärmeväxlare. I fig. 2 syns det avsnitt ur CAS där några uppgifter för en önskad värmeväxlare specificeras. Efter att i andra "stationer" i CAS gett andra uppgifter kommer man till denna "vy". Slutligen kommer programmet att välja ut lämpliga PHE ur en databas på grundval av inmatade data. Inmatningen av information går så tillväga att man stegar sig fram genom de olika inmatningsfälten med hjälp av *tab* och *shift+tab* tangenter. Vid vissa fält ska man själv ange ett värde, medan det vid andra fält poppar upp en meny där val göres.

Förslag till prototyp

I fig. 3 finns huvudfönstret i prototypen. En del inmatningsfält ligger i huvudfönstret medan andra ligger ligger i underfönster som framträder vid vissa operationer som tryck på knappar och bilder. Alla fält i fig. 2 har sin motsvarighet här och en del tillägg har också gjorts. "Program select" anger, till skillnad från "Specify", att programmet själv ska ange exempel på PHE enligt angivna data. Dock ska man här välja mellan olika grundtyper nämligen lödd, sammanfogad med packningar eller (delvis) svetsad (brazed, gasket, welded). Dessa val har betydelse för utseendet av bilden under "Gasket".

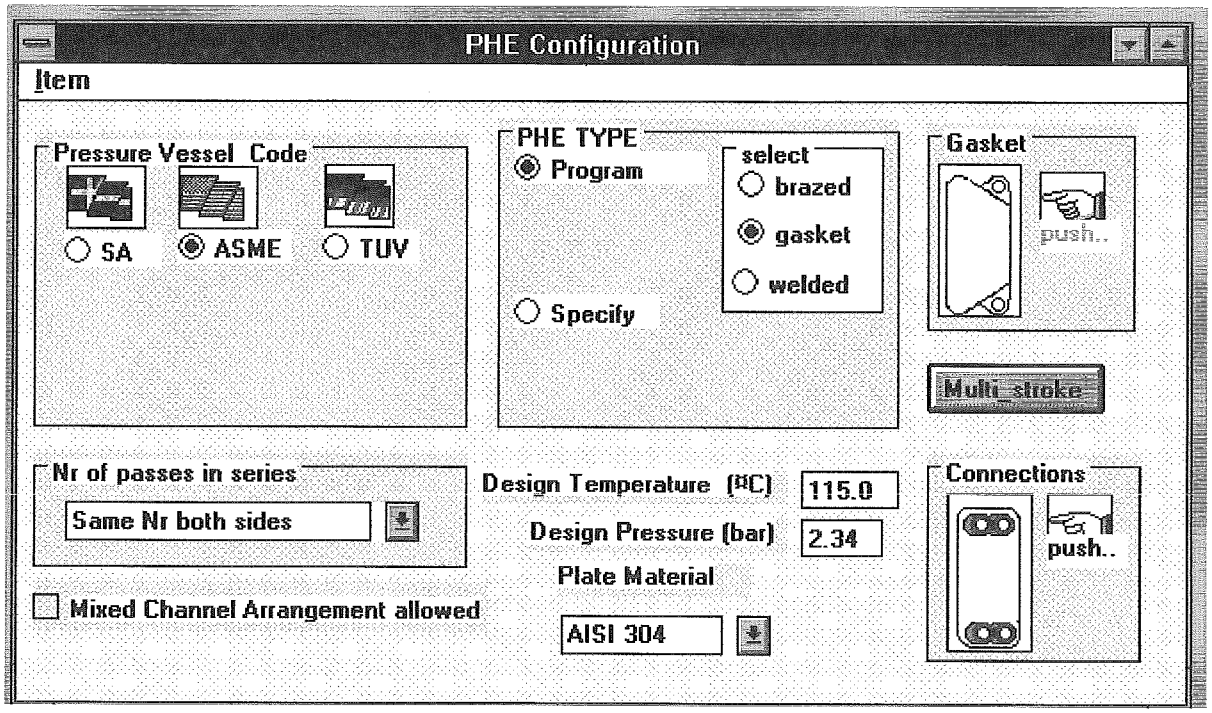


Fig. 3 konfiguration av plattvärmeväxlare

Vid val av "welded" syns här en annan bild på en packning, medan den saknas helt vid "brazed" då packningar inte alls finns inuti lödda plattvärmeväxlare. Klick med mus på packningsbilden innebär att ett underfönster dyker upp, där några specifikationer för packningarna ska anges. Med klick på bilden

"Connection" specificerar man på liknande sätt anlsutningarn till phe´n. Val av flagga anger val av tryckhållfasthetskod enligt resp. lands (Sverige, USA , Tyskland) normer. Vid klick på "Multistroke" kommer det upp ett fönster, där man grafiskt väljer anslutnigar som ska vara förslutna hos vissa typer av phe.

Kunddatabas

Då en försäljare är ute hos en kund kan det finnas behov att få upplysningar om tidigare installationer och detta är också viktigt för en servicetekniker. I stället för att försäljaren ska släpa med sig mängder med pärmar eller att kunden själv måste kunna tillhandahålla uppgifter om modell, tillverkningsnummer etc. genom att i värsta fall gå och damma av skylten på PHE´n, vore det smidigt om försäljaren på sin notebookdator snabbt kunde få fram alla nödvändiga data eller "Information at your fingertips" som det också kallas. Detta är ett uttryck som myntats av Microsoft´s grundare och Vd, Bill Gates. För detta ändamål finns idag ej något program om man bortser från en för minidator med textkommandon baserad databas. Som utgångspunkt har då en ritning till en kundorder använts. Denna innehåller enligt den "informationsanvarige" (se sid 22) all väsentlig information. I denna prototyp, liksom i den föregående, presenteras informationen i huvud- och underfönster (fig. 4 resp 5). Då man t.ex. i huvudfönstret väljer en detalj av översiktsbilden av ritningen med ett musklick, markeras "fokus" på denna med en mörk ram. Vid dubbelklick kommer detaljen att synas i underfönstret "Detail Viewer". Fönstret "Properties ..." dyker upp då man med musen trycker på knappen "Media".

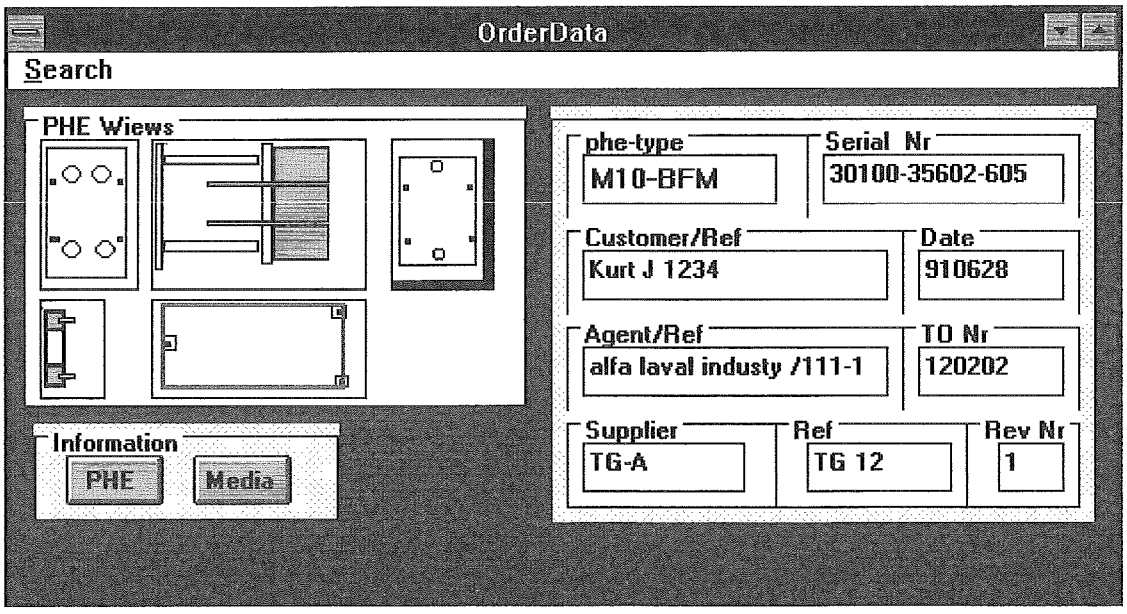


fig. 4 kunddatabas

Här har det minneskrävande bildformatet bitmap använts, vilket har varit enklast att använda. I det slutgiltiga programmet krävs dock ett betydligt mindre minneskrävande format. Det viktiga här är att visa hur uppgifter om mått etc. kan åskådliggöras och skalenliga bilder är inte alls nödvändiga.

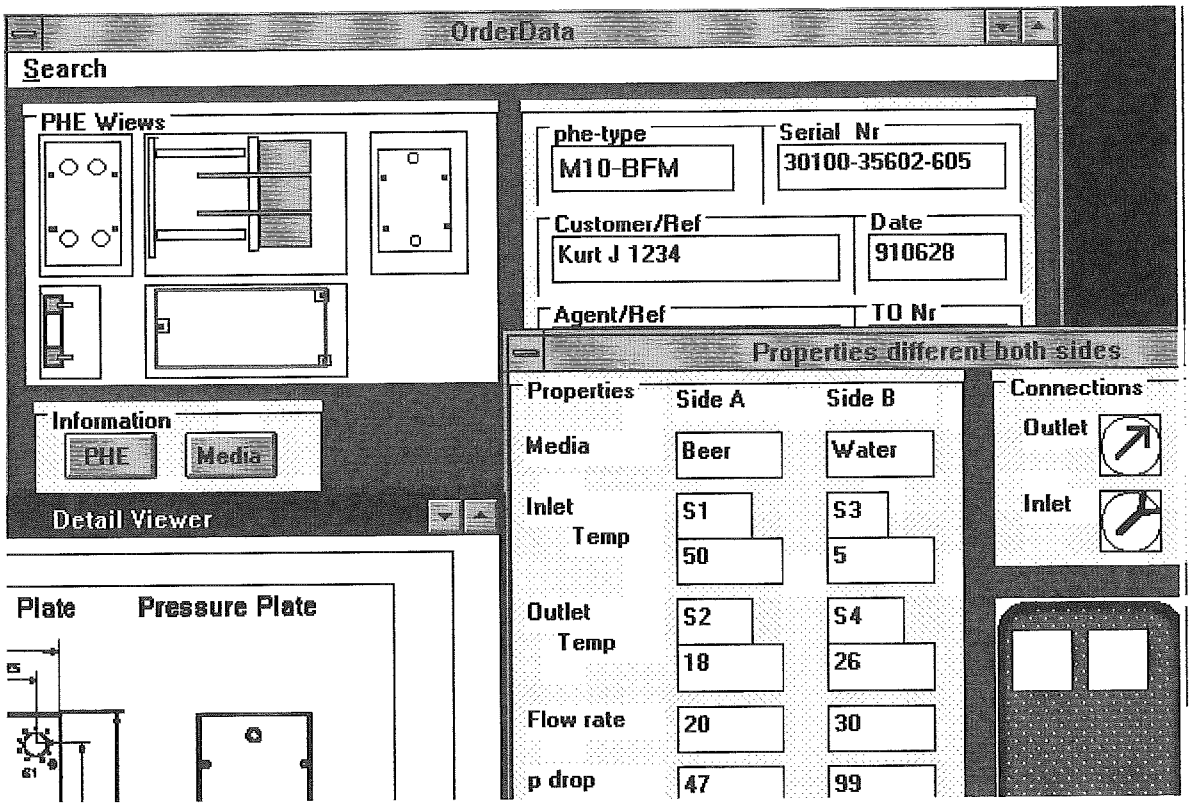


fig. 5 kunddatabas

3.2 Arbetsgång vid framtagning av prototyper

Det är viktigt att förstå hur användaren tänker. I detta fall genomgicks en tre dagars givande kurs om värmeväxlare. Informationen i prototyperna berör nämligen på ett eller annat sätt just värmeväxlare. I denna förutsattes viss förkunskap i t.ex värmeöverföring och strömningslära. I kursen genomgicks principen med plattvärmeväxlare, dess användningsområden, underhåll samt rundvandringar i produktionen.

För varje prototyp har det funnits en person som varit ansvarig för den information som den blivande prototypen kan tänkas innehålla. Den inledande kursen gav en orientering om den tekniska bakgrunden, vilken bildade en god grund för kommunikationen med dessa personer. Dessa personer är dock väl införstådda med informationen,så någon riktig användare som ska lära sig systemet från grunden är denne inte. Försäljaren ute på fältet kan också ha en annan syn på användandet än den som sitter på huvudkontoret. Detta har dock här de

redogöra för bakgrund, visioner, lite om hur den riktige användaren arbetar och tänker samt gett vissa personliga tips på detaljer i prototypen. Efter detta har implementeringen av kod tagit vid och arbetet med en utkast till prototypen påbörjats.

För användargränsnitt avsedda för övervakning av processer har det inom området teknisk psykologi gjorts undersökningar om hur användaren mottager informationen. För beslut vid kritiska situationer är det viktigt att informationen mottages korrekt och snabbt av användaren. Viktiga faktorer är här sådant som gradering av skalor, kombinationer av färger, den hierarkiska lokaliseringen av informationen om all information ej kan presenteras samtidigt etc. Prototyperna här är avsedda för färdiga program som ska användas i t.ex kundkontakter och dessa är här mer ett stöd i kommunikationen. Därför har utkasten här mer gjorts på intuition, även om effektsökeri medvetet undvikits.

Bilder som kan tänkas användas har skapats främst i det enkla programmet Paintbrush som ingår bland standardprogammen i Windows eller med en scanner. Denna scanner har dock varit lite klumpig att använda pga programvaran till den är gammalmodig och bara kan användas i Windows realtids mode som används för applikatoner i Windows tidigare än version 3.0 . Inmatningsfält etc har lagts ut på måfå i fönstret till en början, viss information har också placerats i underfönster vilka dyker upp på skärmen vid operationer i huvudfönstret. Efter detta har utkastet fått struktureras om lite så att det tänkta användandet sett någorlunda rimligt ut samt att den ser "visuellt harmonisk" ut så att den vid första anblicken ska väcka intresse. Detta luddiga begrepp kan innefatta sådant som att olika element (textfält, knappar etc) i en viss logisk grupp t.ex har samma position för vänster- eller ovankant. Dessa grupper har sedan getts ett lämpligt relativt storleks- och avståndsförhållande. Om ett par element läggs till i en grupp kan det ändra gruppens utseende och det kan medföra att man börjar ändra i de andra grupperna osv. så att prototypen slutligen får ett helt annat utseende.

Då utkastet bedömts se någorlunda vettigt ut har det presenterats för den ansvarige. Bedömningen har genomgående varit positiv och inga större ändringar har behövts göras. Omdömena har mestadeles gällt vissa detaljändringar samt förslag till utvidgningar. Det bör tillagas att konceptet med fönster och mus tillsammans med den lätthet färger och bilder har kunnat användas i prototyperna har gett ett bra förstagångsintryck. Personer som inte har varit medverkande i projektet har fått se prototyperna och de tyckt att de sett imponerande ut utan att de haft insikter om funktionen . Bildligt talat har jag fått känslan av att jag med en trollerilåda för barn, "Den lille magikern", åstadkommit illusionsnummer som av publiken ansett vara i klass med vad en professionell trollkarl kan åstadkomma. Att se dessa prototyper och jämföra dem med program skrivna för DOS, har förmodligen varit ett stort steg för betraktarna. Längre erfarenhet av Windows

modligen bättre mognad som innebär mer nyanserade och strukturerade omdömen.

Normerna och riktlinjerna för grundläggande utseende och funktion enligt CUA har varit ganska lätta att följa med de använda verktygen, speciellt då hos Visual Basic. Att studera många program för Windows har varit till stor hjälp, då det inte funnits litteratur tillgänglig som kunnat ge råd för dessa specifika tillämpningar. Människor som sett prototyperna har som sagt varit imponerade och kritik huruvida dessa verkligen uppfyller CUA-normer eller om information verkligen presenteras på ett logiskt sätt, har kommit i skymundan. T.ex är det viktigt att ha ett gränssnitt för tangentbordet, vilket är viktigt för notebook-datorer, men användare har här varit tillräckligt tillfredsställda vad de kan utföra med musoperationer. Detta har för dem i det här sammanhanget varit nytt och intressant.

3.3 Arbetsinsats

I början fanns endast Turbo Pascal för Windows att tillgå. Visual Basic blev tillgängligt drygt en månad senare. Den första tiden ägnades parallellt både åt att sätta sig i problemet med användargränssnitt och prototyper och att sätta sig i och att utforska möjligheterna med Turbo Pascal. Som tidigare nämnt ägnades mycket tid åt att förstå hur man kunde använda detta verktyg. Denna Pascal är annorlunda än standard Pascal. Detta gäller speciellt omfattningen av dess användning som programmeringsspråk för Windows. I de medföljande manualerna finns det typexempel på program med grundläggande uppträdande typiskt för applikationer i Windows. Dessa mallar fanns dock inte vara tillräckliga för utvecklandet av en del ideér av användargränsnitt. Annan litteratur för programmering i Windows med Turbo Pascal fanns inte. För att utforska dessa ideér som inte passade in i de angivna mallarna, fick mycket tid ägnas åt "trial and error" i programmeringen för att många detaljer i prototypen. Den första prototypen tog därför cirka ett par veckor att åstadkomma.

Visual Basic var betydligt lättare att lära in. De tidigare erfarenheterna med att lära sig att förstå Windows underlättade också, och enklare Visual Basic program kunde åstadkommas 2-3 dagar efter det att utforskandet av detta verktyg påbörjats. Även utan den tidigare erfarenheten hade det nog inte tagit speciellt lång tid. Ett första förslag till en prototyp kunde sedan ta så lite som 3-4 dagar att åstadkomma. En av de tidssparande fördelarna hos Visual Basic gentemot Turbo Pascal är att dess utvecklingsmiljö är ett skal som hindrar felaktigheter i programmen att helt låsa Windows. Sådana fel var vanliga i programmeringen med Turbo Pascal. Dessutom är det också enklare att spåra felaktigheter eller "buggar"

under denna utvecklingsmiljö. När programmen sedan väl är utprovade kan man åstadkomma en vanlig EXE-fil som kan användas i andra datorer.

4 Att programmera i Windows

4.1 kort om objekt orienterad programmering

Objektorientering har de sista åren varit ett modeord i datavärlden och det nämns här bl.a därför att objektorienterad programmering är något som underlättar programmering i Windows. I objektorienterad programmering ses programmets exekvering som en fysisk modell som simulerar uppförandet av någon del av omvärlden. Det objektorienterade perspektivet ligger närmare till fysik än matematik. Istället för att beskriva en del av världen med hjälp av matematiska ekvationer,konstruerar man en fysisk modell. Objektorientad programmering imebär att både datastrukturer paketeras till enheter, objekt. Fördelen med denna paketering är att delarna i datasystemet blir väl avgränsade och kan användas som byggklossar. Denna *avgränsning* innebär att den inre strukturen hos objekten kan skyddas. I objektorientering interagerar objekten endast via s.k *meddelanden*. Detta medför att man definiera rena gränssnitt mellan objekten. Ett meddelande talar om vad som önskas, inte hur det ska erhållas.Ett objektorienterat program kan också ses som skådespelare på en scen där programmeraren är regissör och styr replikerna mellan skådespelarna. *Arv* är ett begrepp som innebär att nya objekt kan ärva beteende från redan definierade objekt. Detta är mycket användbart och kraftfullt i återanvändningsyfte men kräver samtidigt ett välstruktuerat förarbete för att man skall kunna dra nytta av de positiva effekterna.

4.2 Programspråk som har utprövats

Programmeringsspråket C tillsamman med Microsoft SDK var länge det enda realistiske alternativet för programmerare i Windows. Det har lång inlärningstid och kräver mycket detaljkunskap om operativsystemet. Att Windows ännu ej är ett riktigt multikörningssystem gör det extra omständligt.Sedan våren 1991 finns den objektorienterade versionen av C, C++ i versioner för Windows. C++ kan tillsammans med ett objektorienterat bibliotek underlätta programmeringen betydligt och avsevärt korta inlärningstiden. Ett annat språk på denna nivå som prövats är Borlands Turbo Pascal för Windows. Denna variant av Pascal har mycket omfattande tillägg jämfört med standardiserad Pascal, bl.a objektorientering. Någon som enbart läst en grundkurs i programmering baserad på standard pascal har alltså svårt att utnyttja alla möjligheter hos Borlands pascal. Med Turbo Pascal medföljer ett objektorienterat bibliotek för Windows. Med detta kan programmeraren ganska enkelt skriva kod för program som då får det grundläggande utseendet hos Windows. Om programmeraren vill göra någonting annat än vad som är naturligt med detta biblioteket blir det dock svårare. För den senaste versionen av detta språk lär dock förbättringar ha gjorts. En jämförelse med Borland C++ plus

Hypercardkonceptet

Det finns programmeringsverktyg utvecklade från begreppet hypermedia. Hypercard är en sådan utvecklingsmiljö som finns för Macintosh och den har tagits fram av Apple, det finns också kloner av denna från andra företag. I Macintoshvärlden är hypercard ett vida spritt verktyg och mycket litteratur har getts ut hur man använder detta. Det finns också motsvarande verktyg för Windows, men dessa har inte alls fått samma spridning beroende på att de i detta system anses vara för långsamma och att de kommer från oberoende, mindre företag och de därför inte kan sprida sina produkter lika effektivt som Apple gjort med Hypercard.

Då Hypercard är mindre intressant för Windows har det ej utforskats. Visserligen har det funnits Macar att tillgå, men det är också meningen att demo-ex av prototypen ska kunna spridas och detta är endast möjligt via PC med Windows. Det bör dock nämnas att Hypercard med kloner anses som intressanta för prototyping av användargränssnitt pga styrkan hoss dess grafiska verktyg. Vem som helst som är van vid Macintosh lär kunna lära sig Hypercard på kort tid.

Smalltalk

Smalltalk är det äldsta rent objektorienterade språket. Det tillåter alltså inte någon form av det man kallar konventionell programmering. Objektorienterade hybrid versioner av C och Pascal tillåter att man blandar konventionell och objektorienterad programmering, vilket gör det lättare att stegvis lära sig det senare. I Smalltalk är däremot allting objekt och detta kan vara svårt att lära sig om man är van vid C eller Pascal.

Smalltalk är bara inte ett programmeringsspråk, det anses också av dess anhängare som en av de bäst väl konstruerade och kompletta miljöer för programmering någonsin. Smalltalk är långsamt därför att det interpreteras under exekvering. De som lärt sig det hävdar att det är mycket mångsidigt för t.ex. prototyping. Dessutom är det spritt, versioner finns för Windows, Macintosh, OS/2 och arbetsstationer.

Actor

Actor är en utvecklimgsmiljö för Windows som ytligt påminner om Smalltalk. Syntaxen påminner dock rätt mycket om de objektorienterade hybridspråken. Det är lika gammalt som Windows och tidiga applikationerna gjordes ofta i Actor.

Det är i likhet med Smalltalk långsamt och en annan kritisk faktor är att språkets fortlevnad endast beror på den ende leverantörens existens.

4.4 Objektorienterad programmering och rapid prototyping

Det finns två sätt att se på objektorienterad programmering. Antingen ser man mer på det som ett sätt att effektivt återanvända programkod eller så ser man på det som ett sätt att göra program som skiljer sig från traditionell programmering. Entusiasmen hos förespråkarna för det senare synsättet är ofta oförstådd av de som är vana vid annan programmeringsteknik.

En anhängare av den senare skolan är Mark Mullin. Han har skrivit boken Rapid Prototyping for Object Oriented systems. Rapid prototyping beskrivs som en process, där specifikationen för en viss programvara utvecklas genom samspel mellan en programvaruspecialist, en klient, och en prototyp. Rapid prototyping används när en klient inte från första början kan definiera kraven för en viss programvara till den grad som krävs för att tillfredsställa traditionella designmetoder.

Han använder sig här av Smalltalk på persondator för att i samarbete med klienten åstadkomma ett program som denne blir nöjd med. Detta är en iterativ, upprepad, process där han börjar med ett utkast till användargränssnitt, presenterar detta, gör modifieringar och går sedan vidare med arbetet med den underliggande strukturen.

Att skapa en bra och avspänd relation med klienten, vilken lockar fram dennes ej helt uttryckliga önskemål, är för honom A och O och det är en mycket viktig förutsättnig för hans metod och det kräver rätt personlighet med lång erfarenhet.

Den slutgiltiga prototyp han åstadkommer i smalltalk överför han sedan till mer effektiv C++ kod. Han anser dock inte att det är ett måste att använda smalltalk, utan han anser att det går bra att använda t.ex C++ om utvecklingsmiljön kring denna upfyller vissa krav. Och med dagens förfinande utvecklingsmiljöer för t.ex C++ programmering i Windows är detta nog fullt möjligt bara samspelet mellan kund och utvecklare av programvara fungerar bra.

5 Visual Basic

5.1 Allmänt

Windows led länge av problemet att det var svårt och tungt att göra Windowsprogram. Microsofts SDK, Software Development Kit, var visserligen både generell och effektiv, men den hade den välkända nackdelen att ett enormt kunskapsområde måste behärskas tillfullo av programmeraren innan ens de enklaste rutiner kunde författas. Ett verktyg för genier med gott om tid och intresse för att pyssla med smådetaljer.

När Borland våren 1991 lanserade en objektorienterad Turbo Pascal för Windows kunde man därför inte vänta sig att Microsoft vände andra kinden till. Visual Basic uppfattades då som ett kraftfullt försök att återta initiativet och det har definitivt inte misslyckats även om man naturligtvis inte bör direkt jämföra den med Turbo Pascal för en del punkter. Utvecklingen har visat att Visual Basic och andra verktyg (av vilka en del dock kräver betydligt mer av användaren) som kommit efter denna att SDK snart kan betraktas som ett dystert minne blott.

Namnvalet Visual Basic är dock litet olyckligt. Det handlar om ett makrospråk för Windows med vissa högnivåprimitiver, dvs avancerade funktioner som användaren inte behöver skapa själv. Visserligen används en utvidgad Basic-kod i procedurerna, men detta tredjegenerationsspråk är så utformat att Pascal-, Modula- och C-programmerare finner sig väl tillrätta. Slamkryparna GOTO radnr och GOSUB radnr finns inte här. Denna Basic skiljer sig alltså markant från den Basic som många t.ex lärt sig att skriva program i på hemdatorer och på gymnasieskolornas minidatorterminaler under första hälften av 80-talet. Denna Basic används dock fortfarande i t.ex mindre krävande mätdatorsystem av mindre dimensioner. Man stöter t.ex på denna i laborationer i kurser hållna av institutionen för Elektrisk Mätteknik.

Högnivåprimitiverna är här så kraftfulla att endast en del av uppgiften behöver skrivas i vanlig kod, nämligen algoritmerna. Det tunga arbetet med att administrera programkörningen, fönstren och alla grafiska faciliteter sköts helautomatiskt av Visual Basic.

Att denna Microsoftprodukt blivit döpt till Visual Basic ska nog närmast ses som en nostalgisk återblick mot Microsofts historiska förflutna. Deras första stora produkt var nämligen en Basic-tolk.

5.2 Detta omfattar Visual Basic

Visual Basic för Windows från Microsoft omfattar följande:

- En sömlös integrerad utvecklingsmiljö för att interaktivt skapa, avlusa, kompilera och underhålla program. Syntaxkontroll och debugger med stegvis eller procedurvis körning samt brytpunkter ingår. Dessutom finns ett "immediate window" där aktuella variabelvärden visas.

- En kraftfull projekthanterare för ett program bestående av ett eller flera formulär. Med formulär avses här det som i den färdiga applikationen betraktas som ett fönster av användaren. I Visual Basic kan man dock inte åstadkomma program med variabelt antal fönster, där antalet bestäms först under körningen.

- En kraftfull grafisk interaktiv formuläreditor, inkluderande de grafiska symbolerna komplett, med färg- och musstöd

- En kraftfull och intelligent menyeditor

- Ett ikonbibliotek. Dessutom kan egna ikoner och grafiska symboler användas, av såväl .BMP-format som .ICO-format och Windows metafiler .WMF

- En mycket påkostad och genomarbetad "tutorial" som snabbt och effektivt lär ut hur man använder Visual Basic.

Det går att klippa ut koden i de små demonstrationsprogram som medföljer och klistra in i egna tillämpningar. Den hypertextliknande hjälpfuktionen är synnerligen omfattande och effektiv.

5.3 Språkliga egenskaper

- Programmen är helt händelsestyrda "event-driven". De händelser som kan utlösa en reaktion är användarens hantering av tangentbord och mus, procedurer kopplade till en timer, samt inkommande DDE-anrop (se fakta om DDE) från andra program.

- Kod kan fogas till varje formulär eller element i formulär.

- Kod kan återanvändas i programmet tack vare en global modul. Denna modul är en ren textfil och det är därför lätt att återanvända programkod genom enkelt klippa-och-klista förfarande. Alla tänkbara händelsestyrda procedurer förtillverkas som tomma.

- Felhanterare

- Dynamiska DLL-länkar, Dynamic Link library, för inlänkning av moduler skrivna i andra programspråk. En av nackdelarna med Visual Basic är ju att det är ganska långsamt. Applikationen kan då bli snabbare genom att vissa tidskrävande algoritmer kan skrivas i C eller Turbo Pascal för att sedan kompileras som en DLL-moduler. DLL-moduler benämns som dynamiska därför att de anropas och länkas till applikationen under programkörning

- Dynamisk dataöverföring, DDE, både till och från andra program. Detta är ett begrepp för överföring av data mellan oberoende program under Windows.

- S.k systemanrop för Windows finns tillgängliga, dock finns inte här dokumentation av dessa.

- Förbindelse finns med "klippbordet". Detta är den speciella fil i Windows som används när man utför klippa-och-klistra operationer. Klippbordet utnyttjas också vid DDE-kommunikation

5.4 Fakta om DDE

Dynamic Data Exchange är i Visual Basic lätt att hantera och fungerar på följande sätt. En länk kan upprättas mellan en klientapplikation, den som anropar, och en serverapplikation, den som svarar på anropet. Länken kan vara en traditionell överföringslänk, en kall länk eller en het länk. Över en het länk skickas data över på nytt varje gång de ändras, utan anmodan eller särskild bevakning av den saken.

Utöver data kan även tangentnedtryckningar eller hela kommandon sändas med LinkExecute till en annan applikation, dvs för den mottagande applikationen har den sändande applikationen rollen av ett tangentbord .

Program med kapacitet för DDE har ett DDE-namn. Excel anropas sålunda under namnet *Excel* och Word under namnet *WinWord*. Flera länkar kan upprättas samtidigt i olika riktningar. Data skickas från ett formulär i Visual Basic som då är server och kan då kanske mottagas av en textbox som då fungerar som klient.

Algoritmerna som finns under en, till textboxen av DDE-anrop, händelsestyrd procedur styr vad som händer.

Med hjälp av små programexempel som kan hämtas från den utmärkta hjälpfunktionen kan man för den ovane åskådaren utföra ganska imponerande trick. Om man har Excel tillgängligt kan man från en mycket enkel Visual Basic applikation starta denna, sända över sifferdata till ett kalkylark samt slutligen skicka ett kommando till Excel som utifrån informationen i kalkylarket åstadkommer ett diagram av detta. Detta lilla enkla trick gjorde att en från början ganska Windows-skeptisk DOS-entusiast blev lite fundersam

DDE gör alltså Visual Basic till ett bra verktyg för prototyping ur den synvinkeln att man kan kompensera för brister i Visual Basic. Man kan t.ex låta Excel ta hand om enklare databashantering i mindre format, som denna klarar, och Excel tar då emot instruktioner från och skickar resultat tillbaka till applikationen gjord i Visual Basic.

6 Slutsatser

Kravet på ett programspråk som ska användas till implementeringen av det färdiga programmet är att det ska åstadkomma programvara som är portabelt (för olika datormiljöer), det ska vara lätt att underhålla och det ska vara välstrukurerat. Den slutliga programvaran ska också ha god dokumentation.

"Industristandarden" är idag C++ och det finns ingen anledning att man för detta ändamål skulle frånga denna. För ett programmeringsspråk som användes som utvecklingsverktyg är kraven andra. Det ska framför allt vara lätta att göra ändringar. Om man i prototypen åstadkommer en välstrukturerad programkod är detta naturligtvis en fördel då ska överföra denna till språket för implementering av det slutgiltiga programmet. Brister i detta är dock inget större problem då man nu vet specifikationen och har tid att reda ut skönhetsfläckar i koden hos prototypen. Andra krav man kan ställa på prototypverktygen är att de i grunden är grafiskt kraftfulla samt att brister kan kompenseras med t.ex DDE kommunikation eller inlänking av moduler skrivna i andra språk.

Visual Basic som verktyg för prototyping var här en stor framgång. Enkla prototyper för Alfa-Lavals ändamål utförda med Visual Basic tog bort all tveksamhet rörande Windows lämplighet och Visual Basic blev snabbt den nya flugan hos personal som på något sätt sysslade med programmering. Det finns kraftfullare (och dyrare) verktyg för prototyping, men Visual Basic var här tillräckligt för att engagera alla och en del progammeringskunniga gjorde så småningom egna utkast med anknytning till detta arbete. Med detta försvann en del av svårigheterna att locka fram information från användare.

Dynamiken hos prototypen är också en viktig faktor att poängtera. Att låta användaren använda prototypen lockar fram mer information ur denna än en overhead-presentation av användargränssnittet. Många intressanta detaljer om hur program kan fungera kommer fram i samma ögonblick som operationer utförs och användaren har beteendet hos prototypen som utgångspunkt för vidare diskussion.

Mycket tid lades ner på att utforska Turbo Pascals möjligheter och denna version fanns vara otillräcklig i detta sammanhang. C++ hade lite bättre möjligheter, men dettaa språks möjligheter var likvärdiga med Turbo Pascal om man jämförde dem med Visual Basic. (De senaste versionerna av dessa är enligt fackpress lite mer lätthanterliga och kraftfullare, men Visual Basic har ju samtidigt uppgraderats).

Visual Basic är billigt i inköp, har mycket kort inlärningsstid och utkast till prototyper tar också kort tid att åstadkomma. Det är t.o.m. möjligt att under kort tid göra betydande förändringar i prototypen medan användaren ser på. Brister finns i Visual Basic, men de resultat man åstadkommer gör att ansvariga har en bättre grund än tidigare när det gäller att specifiera programvara för utomstående mjukvaruspecialister.

Färger och grafiska effekter är något som kan verka imponerande vid första anblicken, men det finns risk att användaren tröttnar på ett spelprogramsliknande utseende. En idé är att man i det färdiga programmet själv ska kunna ställa in graden av detta. Denna kan variera för utbildningsändamål, för tillfälliga eller vana användare, nya eller äldre kunder etc.

En annan aspekt är prototypens roll i ett större sammanhang. En del talar om att framtida program mer kommer att vara användargränsnitt, med en del informationsbehandlande funktioner, som är anknutna till generella databaser eller kan använda sig av väl spridda filformat. Hur användandet av prototyper passar i ett större sammanhang har ej undersökts. Tanken är först och främst att åstadkomma program fungerar tillfredsställande individuellt för bärbara persondatorer, men man ska ej glömma bort visioner hur dessa i framtiden kan utnyttja en gemensam databas. Det vore t.ex. intressant att tänka sig en gemensam databasstruktur för de olika informationssystem som de två PHE-prototyperna i avsnitt tre reprensenterar. Alfa-Laval Thermal bör dock kanske invänta de närmaste årens utveckling inom främst hårdvara och systemmjukvara som förhoppningsvis innebär en bättre integration av olika informatonssystem innan man lägger ner större resurser på detta. Brist på bra standarder är betydande orsak till att vi idag inte kan uppfylla visioner om "information at your fingertips" i större utsträckning. För strategiskt tänkande är det dock viktigt att hålla ett vakande öga på utvecklingen inom området. Målen med detta detta arbete har dock uppfyllts och man har kommit på Windowståget i god tid. Möjligheterna att själv påverka utvecklingen i framställningen av programvara "inomhus" är stora och kvaliteten på det man kan åstadkomma är klart tillfredställande för detta sammanhang. Om något skulle visa sig vara mindre tillfredsställande är det här lätt att kunna återkoppla detta till framställningsprocessen, som man själv har kontroll över. Man vet ju bättre själv hur man har funderat än man vet hur en utomstående (konsult) tänkte. Denna process innefattar bl.a. hur man lockar fram åsikter ur användaren. Denna kan då förbättras och resultatet kan bli bättre nästa gång. Visual Basic har passat utmärkt för detta företag som har begränsade resurser för att låta tillräckligt med personal ha kompetens för att arbeta med mer sofistikerade och betydligt kostsammare utvecklingsmedel.

Alfa Laval Thermal har en decentraliserat organisation och för att en sådan ska överleva bör man vara något restriktiv med de ekonomiska resurserna. Visual

Basic passar här bra, för det är relativt billigt och det är demokratiskt i det avseendet att betydligt fler än ett fåtal specialister kan påverka användandet av detta!

Litteratur

Rapid prototyping, Mark Mullen,1989,Addison-Wesley

Manualer:

Windows, Visual Basic, Microsoft

Turbo Pascal för Windows, Borland

CUA-Standard, IBM