TRIAD Arbetsrapport U1

Krav på IA
Nästa Generation Modellering
Avancerad utbildning för handledare
Katalogprinciper
Uttagssystem
Informationspridning

Rapport Unr 1: Hybris i Unix-miljö, förstudie

Hybris i Unix-miljö – förstudie

Thomas Bill, SISU

Rapporten är skriven i och för TRIAD delprojekt Uttagssystem.

TRIAD-projektet drivs gemensamt av Televerket, Posten, Statskontoret , Ericsson Data Services och SISU, Svenska Institutet för Systemutveckling.
För vidare information kontakta SISU Informationscentrum, tel 08-752 16 00, fax 08-752 68 00,Box 1250, 164 28 Kista.

Innehåll

Hybris i Unix-miljö, förstudie

TRIAD Datorstöd Rapport D nr 1
Hybris i Unix-miljö, förstudie- Arbetsrapport
Thomas Bill

1. Inledning

Syftet med denna förstudie har varit att utreda de tekniska förut-sättningarna för en realisering av Hybris i Unix-miljö. Tyngdpunkten av arbetet har lagts på att skaffa en överblick över de verktyg som för närvarande finns för att bygga grafiska användargränssnitt för Unix och att testa de som ansetts lämpliga. Förstudien har också utrett möjligheten att använda kod från TeleSofts Glue i en Unix-version av Hybris. Testerna har gjorts med utgångspunkt från de egenskaper för verktyg som tas upp i kap 2.1.

Det finns även informationsstrategiska anledningar till en förstudie av det här slaget. Enligt de riktlinjer som dras upp i IP-90 (Policy för informationshantering inom Televerket) bör så stor del som möjligt av ny- och vidareutveckling inom Televerket ske i Unix-miljö. En systemstruktur som bygger på Client/Server konceptet rekommenderas allmänt genom införande av Unix-baserade arbetsstationer. Vidare sägs att standardiserade grafiska gränssnitt bör användas i så stor grad som möjligt och att nya databaser skall byggas med relationsdatabasteknik och SQL-gränssnitt. De olika beslut som fattats om öppna system inom Posten överenstämmer i stort med Televerkets, men finns inte sammanfattade i ett dokument.

Med dessa riktlinjer i åtanke är det naturligt att undersöka hur Hybris skulle kunna verka i en Unix-baserad miljö och tjäna som grafiskt gränssnitt mot både nuvarande och framtida relationsdatabaser.

Förstudien har ingått som en aktivitet inom delprojektet uttagssystem "Hybris" inom TRIAD och utförts av Thomas Bill, med assistans av Peter Rosengren. Dessutom har följande personer deltagit i möten och diskussioner:

Jerzy Kawa, Televerket Data

Örjan Jonsson, Televerket, Pu

Gunnar Boström, Telesoft Sundsvall AB

Marie Helén Andersson, Telesoft Sundsvall AB

2. Utveckling i Unix

Det här kapitlet tar upp några aspekter på Hybris som ställer särskilt stora krav på implementationsmiljön. Kapitlet tar också upp två alternativa ansatser för konstruktion, nuvarande miljö samt behandlar två sk *Style Guides*.

2.1. Vad krävs av ett verktyg för Hybris

Det som framför allt präglar Hybris, jämfört med många andra tillämpningar, är den höga graden av direktmanipulation. Med direktmanipulation menas att användaren kommunicerar med Hybris genom att manipulera olika grafiska objekt med musen. De som använt Hybris vet att nästan alla operationer innefattar någon form av direktmanipulation. Navigering i informationskartor, följa länkar i uppslagsboken och menyval, är alla operationer som innehåller moment av direktmanipulation. En konsekvens av detta är att Hybris innehåller en stor mängd grafiska objekt med olika beteenden, vilket indikerar att Hybris bör utvecklas i en objektorienterad miljö. En objektorienterad ansats är ofta lämplig vid utveckling av tillämpningar med många grafiska objekt med olika beteenden. En annan konsekvens är att Hybris är händelsestyrt, vilket innebär att Hybris hela tiden ligger och avvaktar användarens kommandon och omedelbart reagerar på användarens operationer. En eventuell utvecklingsmiljö för Hybris bör därför innehålla stöd för utveckling av händelsestyrda tillämpningar. Dessutom bör det finnas stöd för att ta fram komplexa dialogboxar, något som Hybris har gott om och som kan vara svårt att konstruera utan bra stöd.

En viktig egenskap hos en utvecklingsmiljö är stöd för att snabbt ta fram prototyper. För att underlätta framtagningen av prototyper bör det också finnas bra stöd för att experimentera med de grafiska objektens utseende.

Nedan följer några egenskaper som en utvecklingmiljö för Hybris bör ha:

- Objektorienterad
- Ett bra programmeringsspråk för att definiera datastrukturer och beteendet hos objekt.
- Stöd för händelsestyrd programmering
- Bra stöd för framtagande av dialogboxar
- Det skall vara lätt att experimentera med de grafiska objektens utseende.
- Stöd för kommunikation med tillämpningar utanför Hybris.

2.2. Motif och Open Look

Eftersom fönstersystemet X windows ser ut att bli en standard inom Unix-världen bör Hybris för Unix implementeras i denna miljö. X windows finns på ett stort antal plattformar och har en mängd fördelar, men det är också känt som ett mycket svårprogrammerat system, vilket gör att valet av verktyg är mycket viktigt.

Om vi utgår från att Hybris skall verka i en X windows-baserad miljö, så finns det framför allt två konkurrerande sk *Style Guides*. En Style Guide är en specifikation av funktionalitet och utseende hos de delar som ingår i användargränssnittet. Den första är Motif ifrån OSF (Open Software Foundation). OSF är en sammanslutning av företag och organisationer, bl.a Televerket, som arbetar för en standardisering inom Unix-världen. OSF grundades för att vara motvikt till Sun, AT&T och dess allierade i Unix International, som står för den andra beskrivningen, Open Look. Open Look finns implementerat i Suns fönstermiljö OpenWindows.

För att realisera gränssnittet används ett sk toolkit. Ett toolkit är ett subrutinbibliotek med procedurer för att rita knappar och andra interaktionsobjekt. Fig.1 visar hur begrepp som toolkit och Style Guide hänger ihop med andra begrepp i X windows.

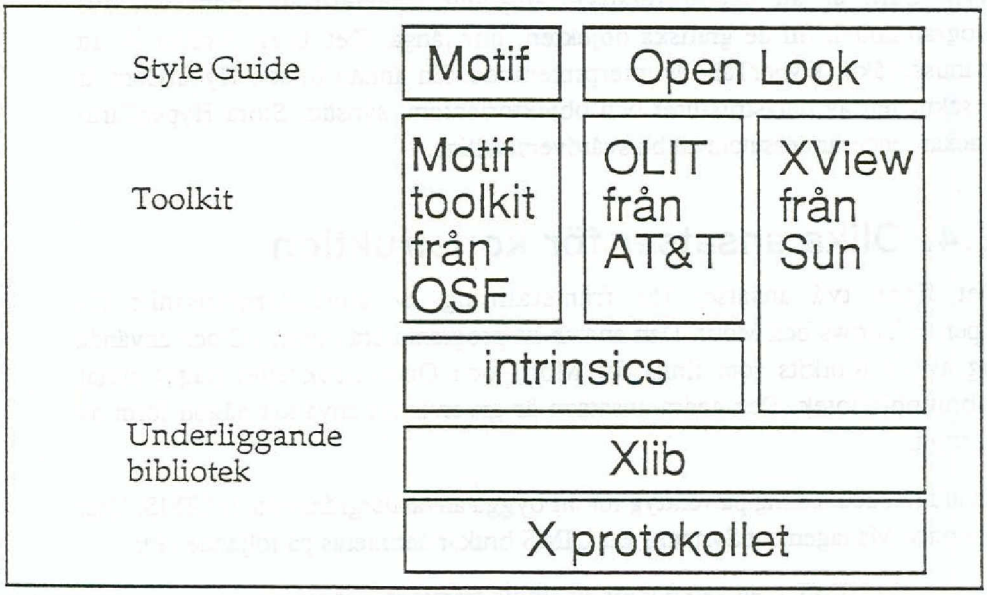

FIG.1 **Sambandet mellan Style Guide, toolkit, intrinsics, Xlib och X protokollet**

Det bör klargöras att begreppen och de olika försöken till standarder inom X windows-världen är en djungel och att den här rapporten inte har någon ambition att försöka reda ut dessa begrepp. Vi utgår istället ifrån att Open Look och Motif är de två huvudkandidaterna.

Det finns några argument som talar för att man bör välja Motif. Bl.a har Televerket tagit fram en standard för hur användargränssnitt skall se ut, TVT-GMMS, Televerkets standard för Grafiskt Människa-Maskin-Språk, som bygger på Motif 1.0 och den äldre standarden TVT-MMS. Det är självklart en stor fördel om Hybris har ett användargränssnitt som är konsistent med övriga tillämpningar. Det bör dock påpekas att det är högst oklart hur pass väl TVT-GMMS har slagit igenom inom Televerket. Posten har inte tagit fram någon standard för användargränssnitt utan prövar för närvarande flera alternativ.

2.3. Nuvarande miljö

Den nuvarande versionen av Hybris är implementerad med HyperCard från Apple, kommunikationen hanteras av MicroPhone II och SQL-generatorn är skriven i C med hjälp av Unix-verktygen lex och yacc. Eftersom SQL-generatorn är skriven i C är den direkt flyttbar till Unix.

Användandet av HyperCard är antagligen en av orsakerna till att Hybris-projektet gått så pass bra. Det är en mycket flexibel miljö som gör det möjligt att ändra i gränssnittet under utvecklingsarbetets gång, vilket ofta är svårt vid traditionell programmering. Presumtiva användare har kunnat komma med förslag på gränssnittets utseende och sedan se resultatet relativt omgående. En brist med HyperCard är att det är relativt långsamt, speciellt då manusen, dvs programkoden till de grafiska objekten, blir långa. Det är ett resultat av att manusspråket HyperTalk är interpreterande. En annan brist i HyperCard är avsaknaden av datastrukturer och objektorienterat synsätt. Stora HyperCard-stackar tenderar dessutom att bli svåröverskådliga.

2.4. Olika ansatser för konstruktion

Det finns två ansatser för framställning av användargränssnitt för Open Windows och Motif. Den ena är att programmera direkt i C och använda sig av de toolkits som finns för Motif och Open Look eller något annat subrutinbibliotek. Den andra ansatsen är givetvis att använda någon form av verktyg.

En allmän beteckning på verktyg för att bygga användargränssnitt är UIMS, User Interface Management Systems. Ett UIMS brukar definieras på följande sätt:

- Ett UIMS är ett verktyg för att stödja framtagandet av användargränssnitt.
- Det fundamentala är att användargränssnittet till viss del kan separeras från den underliggande tillämpningen.

Detta är som synes en mycket generell definition och det finns stora skillnader mellan olika verktyg. En del är att betrakta som rena ritstöd, medan andra innehåller stöd för att definiera strukturen på dialogen mellan användaren och tillämpningen.

En stor fördel är om verktyget stöder framtagandet av prototyper och demonstratorer, dvs det skall gå att bygga upp ett gränssnitt och sedan kommunicera med systemet, även om inte den underliggande tillämpningskoden är skriven. De verktyg som är baserade på en kortmetafor är ofta mycket lämpliga för att bygga prototyper. Med kortbaserade verktyg går det att åstadkomma förhållandevis mycket med ringa programmering. Hur än Hybris implementeras i Unix-miljö, så krävs det programmering av traditionellt slag, företrädesvis i C. Användandet av verktyg för användargränsnittet minskar dock det momentet.

Självklart går det även att programmera gränsnittet på vanligt vis. Programmeringen av gränsnittet görs då med hjälp av något toolkit eller något subrutinbibliotek med grafikfunktioner.

2.5. Tidsuppskattning

Hur lång tid det skulle ta att realisera Hybris med de olika ansatserna, är mycket svårt att förutsäga. Om vi jämför ren programmering med de verktyg som testats är dock följande en mycket grov uppskattning:

Traditionell programmering:	2000-3000h
TeleUse, Devguide, DataViews:	1000-1500h
HyperNeWS, FutureShock:	200-800h

Att använda TeleUse, Devguide eller DataViews skulle ta ungefär hälften så lång tid som att programmera traditionellt och med de kortbaserade verktygen skulle det gå ytterligare något snabbare. Den låga siffran 200h gäller för FutureShock och under förutsättning att stora delar av gränssnittet kan återanvändas.

En närmare tidsuppskattning för tidsåtgången vid ren programmering kommer att göras i förstudien "Hybris till DOS-miljö".

3. Utvecklingsverktyg

Följande kapitel är en genomgång av de verktyg som testats inom ramen för förstudien. De verktyg som ansetts som intressantast får en noggrannare genomgång. Läsare som ej är intresserade av de tekniska detaljerna kan hoppa över detta kapitel. För prisuppgifter, se Appendix A.

3.1. Devguide

Developer's Guide, förkortat Devguide, är Sun Microsystems eget verktyg för att bygga användargränssnitt i OpenWindows-miljö enligt Open Look specifikationen för användargränssnitt. Kärnan i verktyget är en palett med de objekt som finns definierade i Open Look. Från paletten drar man ut de objekt som skall ingå i gränssnittet. När ett objekt placerats specificeras de olika parametrar som är förknippade med objektet, det kan vara färg, storlek och exakt position etc. Här specificeras också den procedur som ska anropas när objektet aktiveras, ett sk callback.

När gränssnittet är klart lagras det på sk GIL-format. GIL står för Graphical Intermediate Language och är ett programspråk för att beskriva användargränssnitt. Eftersom det inte finns något som helst programmeringsstöd för språket bör man undvika att programmera direkt i GIL utan i stället låta verktyg som Devguide generera koden. Med programmet Gxv kan sedan GIL-filen kompileras till C-kod. Gxv genererar också en fil med tomma procedur-kroppar för de callbacks som skapats och en Makefile, för att generera och länka ihop den slutliga tillämpningen.

Själva tillämpningskoden skrivs helt separat och kopplas till gränssnittet via de callbacks som definierats i Devguide.

En nackdel är att det inte går att testa särskilt mycket av gränssnittet ifrån Devguide. Att öppna ett fönster med en knapp kräver att man skriver kod med anrop till Suns toolkit XView, att vissa förändringar görs i Makefile och att ett antal kompileringsflaggor ställs om. Det här omständliga arbetssättet gör att det blir mycket mödosamt att experimentera med funktionaliteten hos gränssnittet.

För att få funktionalitet hos objekten måste man programmera med C och XView. Eftersom Open Look följs helt och hållet finns inga möjligheter att med hjälp av verktyget förändra utseendet på objekten, eller utöka deras funktionalitet. En ekvivalent till XView är Open Look Intrinsic Toolkit, OLIT, som ger möjlighet till egendefinierade interaktionsobjekt. Skillnaden är att OLIT är baserat på X Toolkit Intrinsics, se Fig. 1. Förnärvarande går det inte att använda OLIT i Devguide, men en version med stöd för OLIT lär komma inom kort.

3.2. TeleUSE

TeleUSE utvecklas och marknadsförs av TeleSoft i Linköping. Med TeleUSE går det att bygga gränssnitt enligt Motif och MIT/Athena-specifikationerna. Verktyget finns i dag på ett flertal plattformar och anses allmänt vara bland de kraftfullare på marknaden.

TeleUSE har fyra huvudbeståndsdelar:

- En grafisk editor, VIP, för att bygga upp gränssnittet.
- Programmeringsspråket D, för att beskriva dialogen.
- AIM (Application Interface Model), för att binda ihop tillämpningskod och användargränssnitt.
- UIBuilder, för att generera den färdiga tillämpningen.

3.2.1. VIP

VIP är en grafisk editor för att bygga användargränssnitt. På en palett markeras vilket objekt som skall placeras ut och vart objektet skall placeras. Det här är ett arbetssätt som inte är särskilt bra. Det borde istället gå att dra objekten från paletten till den plats som objektet skall placeras på. De objekt som finns i paletten motsvarar det toolkit som har laddats in. En underlig detalj är att inte ikonerna för objekten är de samma som specifikationernas, utan några egna påhittade, som är mycket svåra att tolka. Precis som i Devguide finns en dialogbox för att specificera parametrar för objekten.

Det finns ytterligare ett sätt att placera ut objekt på skärmen. I ett fönster som kallas Tree Window finns gränssnittet representerat hierarkiskt som ett träd. Genom att flytta omkring objekten i trädet ändras deras placering i gränssnittet. Det sistnämnda sättet att arbeta kan låta något komplicerat, men är faktiskt riktigt bra.

3.2.2. D

D är TeleUSE eget språk för att beskriva dialoger. Beskrivningarna består av regler som aktiveras av olika händelser. När en regel matchas utförs de aktiviteter som regeln föreskriver. Att använda ett särskilt språk för att beskriva dialogen har både för- och nackdelar. En fördel är att dialogbeskrivningarna blir förhållandevis små och lättlästa och att dialogen blir separerad från gränssnitt och tillämpningskod. Det sistnämnda gör att det blir lättare att återanvända gränssnittets olika delar. En nackdel är att det blir ytterligare ett språk för utvecklarna att lära sig.

3.3. DataViews

DataViews är ett verktyg från amerikanska V.I. Corporation och kan användas både för att bygga prototyper och färdiga applikationer. I Sverige marknadsförs och säljs DataViews av Kasemen Software AB i Göteborg. Verktyget finns

implementerat på de vanligaste plattformarna. Särskilt väl lämpar sig verktyget för att bygga gränssnitt mot tidskritiska tillämpningar, som processövervakningsystem.

I DataViews skiljer man på gränssnittet och tillämpningskoden. Gränssnittet går att utveckla helt för sig självt. Med hjälp av gränssnitts-editorn DV-Draw är det lätt att rita upp gränssnittet och simulera dialogen. När gränssnittet stabiliserats kan tillämpningskoden kopplas till gränssnittet.

Kärnan i DataViews består av C-biblioteket DV-Tools. DV-tools kan jämföras med de toolkits som finns för olika X windows-baserade system. DV-tools har dock betydligt större funktionalitet. DV-tools innehåller ett stort antal funktioner för kommunikation mellan gränssnitt och program. Kopplingen mellan gränssnitt och tillämpningskod måste till största del göras för hand.

3.3.1. DV-Draw

DV-Draw är en editor för att rita såväl gränsnittslayouter som knappar och andra interaktionsobjekt. Ritfunktionaliteten påminner om de bättre ritprogram som finns i Macintosh- och PC-världen. Det går att rita linjer, rektanglar, polygoner, cirklar, ellipser etc och att skriva text med olika typsnitt. Det går att gruppera och justera objekt på en mängd sätt. Verktyget är byggt för att kunna hantera godtyckliga färgpaletter.

Allt som skapas med DV-Draw är vyer. Vyer kan importera vyer, som då kallas drawings. På så sätt går det att rita upp sina knappar och andra interaktionsobjekt och sedan importera dem in i sitt gränssnitt.

Det finns ett stort antal färdiga interaktionsobjekt, sk input-objekt. Förutom DataViews egna objekt finns också ett antal Motif-objekt. I senare versioner kommer DataViews också att innehålla Open Look- och Athena-objekt. Tyvärr följer inte källkoden till dessa med, vilket gör att inte går att ta ett färdigt input-objekt och modifiera dess beteende. Däremot går det att ändra utseende på objekten.

Men DV-draw är inta bara en editor, det är också ett verktyg för att bygga prototyper. Utan att behöva programmera går det att ge viss funktionalitet åt objekten och simulera den dialog som konstruktören tänkt sig. När en knapp har ritats upp och sedan importerats ger man den dess önskvärda funktionalitet. Tyvärr följer ej funktionaliteten med från ursprungsvyn, vilket är ett resultat av att DataViews inte är objektorienterat. Bristen på objektorienterat synsätt får även negativa konsekvenser på andra sätt.

Det finns flera sätt att koppla data till sina vyer, bl.a genom vanliga unix-filer, Unix-processer eller interna funktioner. Med en funktion som kallas FDSeval finns möjlighet att manipulera inkommande värden med uttryck skrivna i C-syntax. Godtyckliga C-program kan användas för att komma åt och manipulera data.

Om en unix-process skall generera värden åt en variabel, definieras variabelns värde som processens utdata. När sedan gränssnittet provkörs startas processen automatiskt. Detta är mycket användbart när man vill simulera en dialogsituation.

För att visa dynamiska förlopp kan DataViews funktioner för dynamik användas. Med dynamikfunktionen kan grafiska objekt fås att ändra utseende och placering vid olika tröskelvärden eller vid olika händelser. Med dynamik-funktionen är det mycket lätt att bygga animationer.

Utdata kan presenteras med ett sextiotal fördefinierade diagramtyper. När en diagramtyp valts kopplas värdet på axlarna till de önskvärda variablerna. Variablerna kopplas sedan till de önskade datamängderna, vilket gör det mycket lätt att presentera dynamiska förlopp i diagramform.

3.3.2. DV-Tools

DV-tools är ett stort C-bibliotek, som kan jämföras med de toolkits som finns för X windows. DV-Draw har skrivits med DV-tools. För att rita grafer finns ett antal funktioner för att rita figurer av typen objekt-relation-objekt, vilket är av intresse för en implementering av Hybris.

3.3.3. Händelsehantering

När väl gränssnittet skall kopplas till tillämpningskoden så sker det med

C-programmering. Ett visst stöd fås i form av ett programskelett för anrop av de vyer som definierats i DV-Draw. Kopplingen gränssnitt-tillämpningskod är tyvärrr allt för svag, ett problem som de flesta liknande system lider av.

3.4. HyperNeWS

HyperNeWS från det skotska forskningsinstitutet The Turing Institute är en forskningsprodukt som utvecklats sedan 1987. Verktyget går endast att köra under Open Windows, vilket är en stor nackdel. Systemet är ett av de första i Unix-världen som använder sig av samma kortmetafor som HyperCard ifrån Apple och arbetssättet är ungefär det samma. Som nämts tidigare är kortbaserade verktyg ofta lämpliga för att ta fram prototyper, så även HyperNeWS.

En nackdel med HyperNeWS är att manusspråket är PostScript, som kan vara svårbemästrat. En annan nackdel är att systemet inte stödjer de riktlinjer för användargränssnitt som definieras av Open Look, trots att de tillämpningar som tas fram bara kan köras i Open Look-miljö.

HyperNeWS distribueras gratis från Turing institutet, vilket innebär att man inte garanterar någon support på systemet och att dokumentationen är bristfällig.

HyperNeWS passar bra till att bygga enkla prototyper med, men avsaknandet av support, den bristfälliga dokumentationen och det faktum att PostScript används som manusspråk, gör att systemet är olämpligt att bygga kommersiella system med.

3.5. FutureShock

Progress Software säljer och utvecklar 4GL-språk och relationsdatabaser. Sedan några år tillbaka utvecklar man även FutureShock.

FutureShock är en HyperCard liknande utvecklingsmiljö för ett antal olika plattformar, för närvarande Sun3, Sun4 och DECstation 3100. Enligt Progress kommer även systemet att finnas för HP/Apollo, RS/6000 och PS/2. Progress kommer att överväga en portering till Macintosh när System 7.0 lanseras. Tillämpningar utvecklade på en plattform är direkt flyttbara till de andra.

Miljön är uppdelad i två nivåer, där den enklare är identisk med HyperCard. I den andra nivån försöker FutureShock råda bot på några av de begränsningar som finns i HyperCard.

3.5.1. HyperCard-kompatibel

HyperCard 2.0 stackar är kompatibla med FutureShock genom formatet HIFF, (Hypermedia Interchange File Format). HIFF ser ut att bli en de facto standard för konvertering av stackar mellan HyperCard-liknande verktyg. Med Heizer Software´s produkt ConvertIt konverteras HyperCard-stackar till HIFF. Efter konvertering av en HyperCard-stack med ConvertIt läser FutureShock in HIFF-filen och bygger en egen stack med identiskt utseende.

Ett problem som dyker upp under alla konverteringar av HyperCard-stackar är externa funktioner (kodresurser). FutureShock erbjuder ingen direkt lösning på problemet. I stället minskas behovet av externa funktioner genom att erbjuda en objektorienterad miljö med programmerings-möjligheter utöver manus-språket FutureTalk, som är direkt kompatibelt med HyperCards HyperTalk. De klassbibliotek som medföljer och möjligheten att använda "riktiga" programmeringsspråk i FutureShock gör att behovet för externa kommandon inte är lika stort som i HyperCard. Vid behov finns det finns dock möjlighet att anropa externa funktioner skrivna i godtyckligt programmeringsspråk.

3.5.2. Programspråk

FutureShock erbjuder två nivåer för programmering. Den grundläggande är FutureTalk som är helt kompatibel med HyperTalk . När inte FutureTalk räcker till går man över till den andra nivån, som kallas FutureChoice och innehåller två språk, FutureChoice-C och FutureChoice-Lisp.

FutureChoice-C är en förenklad variant av ANSI-standarden med tillägg för objekthantering. Språket är förenklat på så sätt att pekarhanteringen har försvunnit. Utvecklarna menar att denna förenkling gör FutureChoice-C mer lättanvänt och att ett sant objektorienterat språk skall dölja pekarexercisen för programmeraren genom att systemet hanterar pekarna till objekten. FutureChoice-Lisp är i det närmaste identiskt med CLOS (Common Lisp Object System), som också är en ANSI standard. En trevlig egenskap är att uttrycken i de båda språken lagras som evalueringsträd, vilket gör det möjligt att transparent växla mellan de båda språken.

3.5.3. Objektorienterat

FutureShock är objektorienterat i en vidare bemärkelse än HyperCard. Klasser och metoder definieras i FutureChoice-C eller i FutureChoice-Lisp. Verktyget stödjer såväl enkelt som multipelt arv. Bindningen sker dynamiskt och det finns möjlighet att använda sk *prototyping*, dvs det går att använda instanser som förebild vid skapande av nya instanser. Alla objekt som skapas i FutureChoice-nivån kan användas från FutureTalk, vilket gör det möjligt att endast göra klassdefinitioner i FutureChoice och sedan programmera i FutureTalk. Dessutom finns ett stort klassbibliotek med många olika typer av objekt, allt i från grafiska objekt till databas objekt. Detta är en av de stora fördelarna jämfört med HyperCard; utvecklaren är inte låst vid de fördefinierade objekten utan kan fritt skapa objekt med nytt utseende och beteende.

Objekt kan lagras på fyra olika sätt, antingen som lokala objekt som utsätts för garbage collection, lokala objekt som *ej* utsätts för garbage collection, lagringsbara objekt (instanser lagrade på skivminnet) eller delade objekt. De två första behöver inga kommentarer men väl de två sista. Objekt kan lagras i sk *buckets*, som ligger på skivminnet. En bucket kan vara personlig, eller delas mellan flera användare. Delade objekt kan delas mellan flera användare över nätverket. Den sista typen av objekt ger stöd för att bygga groupware-tillämpningar. För att hitta i klasshierarkin finns en klassbläddrare (eng. classbrowser) som påminner mycket om SmallTalks bläddrare. Dessutom finns en instansbläddrare för att undersöka instanser.

En intressant notering är att nästan hela FutureShock är skrivet i FutureShock självt och att systemet är helt öppet, vilket medför stora möjligheter till anpassningar.

3.5.4. Integrering av externa databaser

I FutureShock finns möjlighet att integrera externa databaser, vilket gör det möjligt att skriva SQL-kod direkt i FutureTalk eller FutureChoice. Tabeller, tupler, attribut och vyer kan hanteras som vilka objekt som helst. För närvarande stödjer Progress sin egen relationsdatabas samt Oracle. Företaget har planer på att stödja Informix, DB2 och RDB. Den öppna miljön gör att det inte är svårt att definiera klasser som stödjer access av andra typer av relationsdatabaser.

3.5.5. Motif och Open Look

Den nuvarande versionen av FutureShock bygger på X windows och använder sig av Open Look eller Motif, vilket gör det möjligt att bygga stackar med objekt som använder sig av dessa toolkits. Det finns även möjlighet att importera andra toolkits som är baserade på X Toolkit Intrinsics. Använder man sig av ett Motif-textfält kommer fältet helt att bete sig enligt Motif-specifikationen, vilket är en fördel om tillämpningen skall samexistera med andra Motif-baserade tillämpningar.

3.5.6. FutureMachine

Verktygets kärna är FutureMachine, som är den virtuella maskin som FutureShock körs i. I och med att man valt denna ansats räcker det att portera

FutureMachine, när man vill portera FutureShock. Nackdelen är att systemet riskerar att bli långsamt.

Genom att verktyget använder sk *threads* är det möjligt att köra flera FutureShock-tillämpningar samtidigt. I Unix-miljö körs då alla tillämpningarna i samma process.

4. Databasaccess i Hybris

Som en del i förstudien har det undersökts hur kod från TeleSofts GLUE skulle kunna användas för databasaccess av Hybris i Unix-miljö. I den nuvarande versionen sköts kommunikationen via modem. Frågan skickas och svaret hämtas på batch-orienterat vis. Den nuvarande lösningen har erbjudit en enkel och stabil lösning, om än något långsam. GLUE accessar databasen över nätverket, en lösning som är snabbare och gör det möjligt att plocka fram resultat successivt. Glue har hittills använts mot databaser på Sun- och Apollo-maskiner samt på VAX/VMS.

Hybris SQL-generator är skriven i C och skapad med Unix-verktygen lex och yacc, vilket gör att den är snabb och framför allt portabel. Efter diskussioner med TeleSoft i Sundsvall har vi kommit fram till att det bör vara möjligt att byta ut den nuvarande kommunikationsmodulen i Hybris mot Glues. Gränsnittet mellan Glues SQL-generator och dess kommunikationsdel, är detsamma som hos Hybris, nämligen vanlig SQL. Det skall dock påpekas att någon test ej gjorts, eftersom det har ansetts ligga utanför förstudiens ramar.

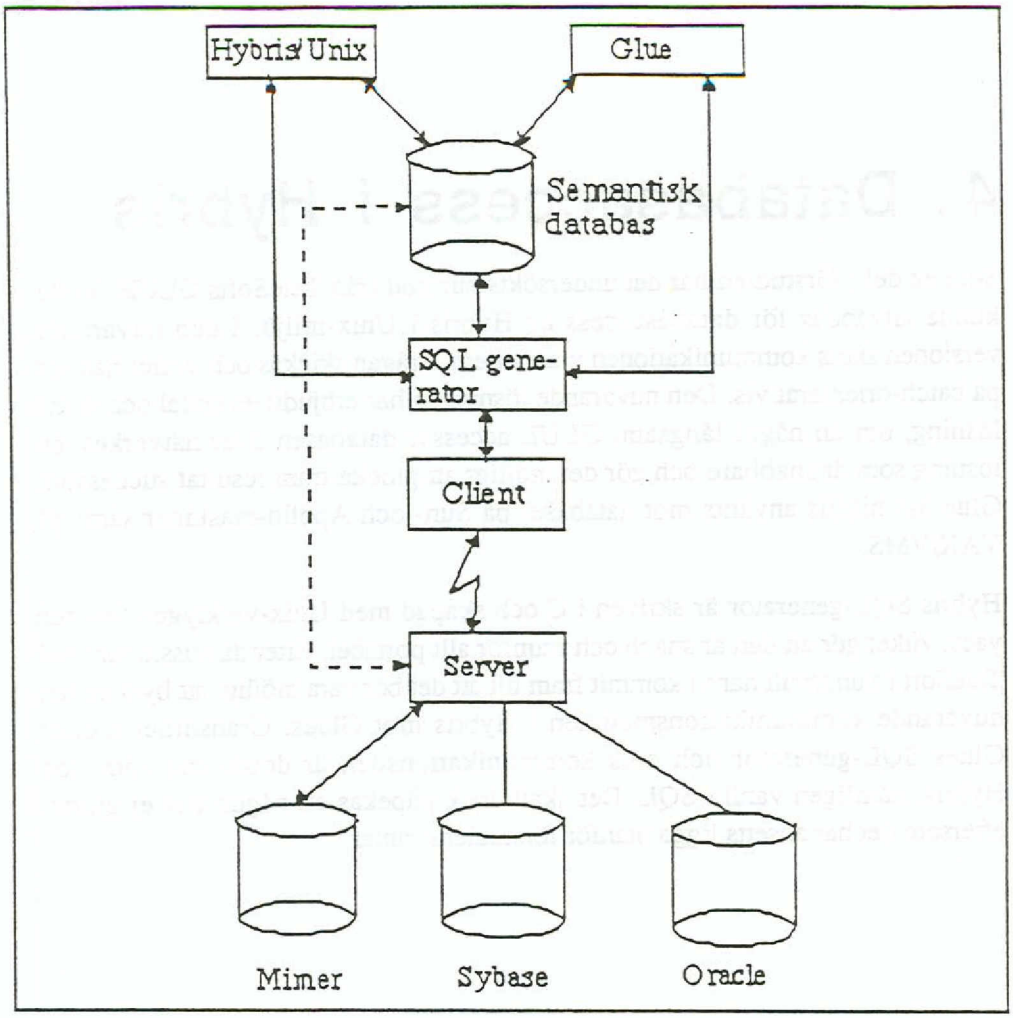

FIG. 2 HYBRIS MED GLUES DATABASMEKANISMER

Fig. 2 visar hur Hybris skulle kunna arbeta i en client/server miljö med hjälp av Glues modul för kommunikation med databaserna. Även Glue finns med på figuren för att visa hur Hybris och Glue skulle kunna samexistera och arbeta med samma semantisk databas och SQL-generator.

5. Slutsatser

Det verktyg på marknaden som idag bäst uppfyller de önskvärda egenskaperna i kap 1.1 är TeleUSE. De andra verktygen har förvisso possitiva egenskaper och uppfyller en del av egenskaperna, men för en implementering av Hybris lämpar sig TeleUSE bäst. Vi rekommenderar dock att man väntar med en realisering av Hybris i Unix-miljö tills verktyg som FutureShock blir tillgängliga. Att använd FutureShock skulle innebära en väsentlig tidsbesparing, eftersom delar av den nuvarande versionen skulle kunna användas.

En nackdel med att använda en helt ny typ av verktyg som FutureShock är att varken posten eller Televerket har byggt upp någon kompetens för att underhålla en applikation utvecklad med verktyget. Här har en traditionell lösning med C-programmering fördelar, eftersom sådan kompetens finns inom de flesta organisationer. Å andra sidan visar erfarenheter från HyperCard-versionen av Hybris att det inte varit några större problem för Televerket att förvalta Hybris.

För närvarande kommer hela tiden nya verktyg för framställning av grafiska användargränssnitt ut på marknaden, varav en stor majoritet är mycket lika Devguide. Ett exempel är VUIT från Digital för att bygga Motif gränssnitt. Trots att de flesta verktyg är mycket lika varandra, finns det anledning att hålla ögonen öppna efter intressanta verktyg.

Med reservation för att inga försök har gjorts, tror vi att Glues databasmekanismer skulle kunna användas av Hybris i Unix-miljö..

Här följer en kort sammanfattning av de intryck verktygen har givit under framtagandet av rapporten och hur lämpliga de är för en implementering av Hybris. För ungefärliga prisuppgifter hänvisas till Appendix A.

Devguide

Det är mycket lätt att placera ut objekt i enstaka fönster. Men tyvärr krävs det ganska snart omfattande programmering för att få någon som helst sorts funktionalitet. Det krävs djupa kunskaper i toolkit-programmering och X windows för att åstadkomma något.

Eftersom verktyget bara stödjer Open Look går det ej, åtminstone inte i den nuvarande versionen, att använda egendefinierade objekt med verktyget. Ett annat minus är att dokumentationen är mycket skral. TeleUSE och Devguide är de verktyg i rapporten som liknar varandra mest. TeleUSE måste dock på grund av större flexibilitet anses vara det bättre av dem.

TeleUSE

Det här verktyget är utmärkt för att bygga dialogboxar och andra statiska delar av ett användargränssnitt med. Förutom större flexibilitet gör möjligheten att specificera dialogen och möjligheten att använda olika toolkits TeleUSE till ett kraftfullare verktyg än Devguide.

TeleUSE skulle kunna ge stöd för framtagning av dialogboxar och de statiska delarna av Hybris, men informationskartorna och navigeringen i dessa skulle då att behöva skrivas för hand.

Eftersom inte FutureShock ännu finns på marknaden, så måste TeleUSE anses vara det verktyg som är mest lämpligt, om en portering skulle göras idag.

DataViews

DataViews är ett mycket imponerande system, men det märks att det har några år på nacken. Utåt ser det helt modernt ut, men avsaknaden av ett objektorienterat synsätt slår igenom. För att bygga gränssnitt mot stora realtidssystem och stora industriella tillämpningar finns antagligen inget bättre, men för att bygga ett verktyg som Hybris, är det för stort och tungt. Helt klart är också att priset avskräcker.

HyperNeWS

Verktyget passar utmärkt till att bygga små enkla prototyper för att illustrera idéer med, men är olämpligt för att bygga system som skall användas i produktion. Eftersom verktyget är gratis så garanteras inget underhåll av systemet. Som utvecklingsmiljö för Hybris kommer FutureShock att vara helt överlägset HyperNeWS.

FutureShock

FutureShock är fortfarande ett verktyg under utveckling. Progress räknar med att ha en beta version ute på test till mitten på sommaren 91. Det blir då ett litet fåtal beta testare med mycket speciella tillämpningar. Med stor sannolikhet kommer SISU att bli betatestare, och då göra försök att portera Hybris till Unix-miljö med FutureShock. Den utvecklingsversion som testats var mycket lovande ut, men pga av att debuggern var aktiv var den långsamt. Det är därför svårt att säga något om framtida prestanda.

Med de egenskaper som FutureShock har, är det ett mycket intressant verktyg. Ett frågetecken är dock tidsaspekten. När släpps verktyget ut på marknaden? När beta-testerna kommer igång framåt sommaren 91 bör vi kunna få en fingervisning om det.

Appendix A

Appendix A innehåller ungefärliga priser för verktygen enligt leverantörerna.

Devguide

Sun Microsystems Svenska AB
Hemvärnsg 9
171 54 Solna
Tel 08/ 705 3000
Utvecklarlicens:
1st≤ 2.300:-/st
Runtimelicens: Nej

TeleUSE

TeleSoft
Box 35
123 21 Farsta
Tel 08/ 713 4200
Utvecklarlicens:
1st 72.000:-/st
- 5st. 61.500:-/st
6st ≤ 48.000:-/st
För första licensen tillkommer 5.800:- för distribution och dokumentation.
För service tillkommer 12.000:-/ st oavsett antal licenser.
Runtimelicens: Nej

DataViews

Kasernen Software
Kasemtorget 6
411 18 Göteborg
Tel 031/ 17 02 80
Utvecklarlicens:
1st 210.000:-/st
2st ≤ 36.000:-/st
Runtimelicens:
1st 30.000:-/st
2st ≤ 10.000:-/st

HyperNeWS

The Turing Institute
NeWS Development
Georg House 36 North Hanover Street
Glasgow G1 2AD
United Kingdom
Dator-post: newsdev@turing.ac.uk
HyperNeWS kan fås genom filöverföring från *lth.se*
Utvecklarlicens:
Gratis
Runtimelicens: Nej

FutureShock

Progress Software Svenska AB
Lumavägen 6
120 31 Stockholm
Tel 08/ 643 93 00
Utvecklarlicens:
Ej bestämt
Runtimelicens: Ja

TRIAD utvecklar IA

Televerket har just tagit första steget in i sin nya IA-organisation och Posten håller på att bygga upp sin nya DA-organisation. Båda organisationerna har sett nyttan att inför 90-talet gå vidare tillsammans i TRIAD-projektet som drivs tillsammans med SISU. Statskontoret deltar också i projektet för att på sikt kunna föra ut nya synsätt och hjälpmedel inom den civila statliga sektorn.

Ericsson Data Services deltar med tyngdpunkten i den del som handlar om att utveckla kompetenta modelleringsledare, delprojektet "Avancerad utbildning för modelleringsledare".

Modelleringsmetoder är centrala i bedrivandet av verksamheten inom informationsadministrationen. Därför arbetar ett delprojekt med utvecklandet av "nästa generation modelleringsmetod" som skall sättas i händerna på informationsadministratören. Siktet är att fördjupa och bredda dagens modelleringsmetoder och där hämta in kunskap från pågående forskning och utveckling internationellt. (faktaruta om IAS91).

Som stöd för informationsadministrationen behövs verktyg. Inom TRIAD arbetar man där inom två områden, kataloger och verktyg.

Delprojektet kataloger arbetar dels med att utforma den informationsmodell som måste kunna täckas av en katalog, dels med att granska och följa utvecklingen av produkter inom området t ex IBM:s "Repository" och Digitals "CDD". Dessutom följer man standardiseringen internationellt kring IRDS. För parterna i projektet liksom för andra organisationer är detta ett tungt område både vad gäller kommande investeringar ekonomiskt och vad gäller kompetenta resurser för en kommande övergång till "repository-världen". - Det inledande skedet syftar till att bygga upp en kunskapsplattform, som sedan kommer att kunna utnyttjas för kravställande och planering och genomförande av övergång från dagens kataloghantering till morgondagens.

Den andra verktygshanterande delen inom TRIAD-projektet, delprojektet "verktyg för informationsadministration", syftar till att ta fram verktyg för uttag och dokumentering av modeller. Betoningen ligger på människa datorgränssnitt och i första skedet görs utveckling av HYBRIS-gränssnittet med prototyper för Posten och för Televerket.

För att hålla ett helhetsperspektiv på projektets delar och för att ha inpassningen av funktionen Informationsadministration i organisationens övriga verksamhet arbetar delprojektet "Krav på IA". I delprojektet arbetar man dels med att kartlägga dagens krav på dataadministration och projicera till morgondagens krav på IA. Dessutom skall man skapa en bild av IA-verksamhetens innehåll och organisation. Från detta i sin tur ställer man krav på övriga delprojekt. Vilka krav skall ställas på kompetens, metoder, hjälpmedel typ kataloger och gränssnitt?

TRIAD projektet är stort

Budgeten för TRIAD-projektet löper på 10 MSEK per år under en treårsperiod som startar vid kalenderåret 1991 års början och som alltså beräknas avslutad vid utgången av 1993.

TRIAD-projektet är ett tillämpningsprojekt

Det innebär att parterna, Televerket, Posten, Statskontoret, EDS och SISU går in med såväl persontidssatsningar som ekonomiska och att STU, Styrelsen för Teknisk Utveckling, bidrar med ett ekonomiskt tillskott som svarar mot ungefär 40 % av den insatta persontiden.

Öppet för fler deltagare

Parterna i TRIAD-projektet vill gärna öka tempot och bredda perspektivet och vill därför gärna ha fler parter in i projektet. Dessa parter får då enligt SISU:s tårtprincip "betala för en tårtbit, men ät hela tårtan", tillgång till projektets resultat med en insats som ger stor "price performance".

Nya deltagare kan gå in i hela projektet eller i det eller de delprojekt som verkar intressantast. En förutsättning är att man framförallt är beredd att satsa kompetent personal. För de flesta intressenter bord detta vara ett utmärkt sätt att driva personalutveckling för personer t ex inom DA-området, samtidigt som man bygger upp beredskapen inför 90-talets IA-verksamhet.

Kompetensutveckling viktigt resultat

En viktig effekt för parterna av deras medverkan i TRIAD är kompetensutveckling. Man satsar på att ta in personer som så småningom eller redan idag arbetar med DA och IA för att ge dem en djup och "frontlinje"-mässig kompetens. Detta skall utnyttjas när man successivt för in resultaten i den egna organisationen. Projektdeltagarna har alltså en viktig roll som kunskapsförmedlare i den egna organisationen. Dessutom ger projektarbetet deltagarna tillfälle till en egen utveckling inom det professionella området som är unik.

Informationsspridning

Det sjätte delprojektet "Informationsspridning" har till uppgift att sörja för att i första hand parterna men också SISU:s övriga intressenter successivt kan följa och tillgodogöra sig resultat från TRIADprojektet. Seminarier, rapporter och referensgruppsverksamhet är led i den verksamheten.