TRIAD Rapport K 23

XLII
– en öppen och flexibel utvecklingsmiljö

Dan Nyström

TRIAD-PROJEKTET DRIVS GEMENSAMT AV TELEVERKET, POSTEN, STATSKONTORET, ERICSSON OCH SISU, SVENSKA INSTITUTET FÖR SYSTEMUTVECKLING. FÖR VIDARE INFORMATION KONTAKTA SISU, TEL 08-752 16 00, FAX 752 68 00, ELECTRUM 212, 164 40 KISTA.

Innehåll

1 Förord

Den här rapporten om öppenheten och flexibiliteten i Intersolvs utvecklingsmiljö XLII har som huvudsyfte att ge stöd till Postens DA-verksamhet vid val av interimistisk resurskatalog och är därför inte helt neutral i alla stycken. Men trots att rapporten är framtagen för att kunna användas i Postens DA-verksamhet kan den vara intressant också för andra. Studien av XLII-miljön genomfördes hösten 1992.

2 Bakgrund

2.1 CASE-verktyg

Under 1980-talet utvecklades datorstöd för att stödja stora delar av systemutvecklingen. Datorstöden kallas i dagligt tal för CASE-verktyg (Computer Aided Software Engineering). Eftersom systemutveckling är en komplicerad process blir också verktygen, eller snarare utvecklingsmiljön, komplicerad.

Hur kan då en utvecklingsmiljö se ut? Nedan beskrivs tre vanliga miljöer.

1. *Ett integrerat verktyg*

Utvecklingsmiljön består av ett enda verktyg som stödjer hela utvecklingsprocessen. Det är mycket sällan som *ett* verktyg från en leverantör kan ge stöd för en organisations hela systemutvecklingsprocess. Det gör att organisationen i de flesta fall får anpassa sig efter verktyget. Dessutom blir man beroende av leverantörens utveckling av verktyget.

2. *Bryggor mellan verktyg*

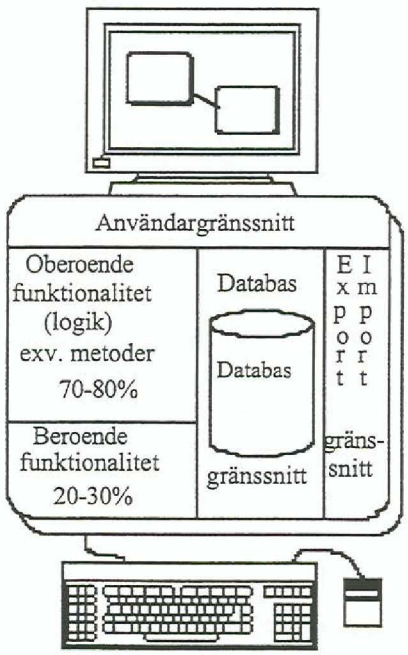
Miljön innehåller flera verktyg och informationen flyttas mellan verktygen med hjälp av export- och importfunktioner. I den här typen av miljö kan man få problem med att ta fram nya versioner av bryggor i takt med att verktygen förändras. Men det största problemet i en stor organisation är att administrera flyttning av information mellan olika verktyg. Till slut blir situationen omöjlig att hantera. En sådan utvecklingsmiljö *kan* fungera om den består av två eller högst tre olika verktyg med bryggor.

3. *En integrerad utvecklingsmiljö*

I en integrerad utvecklingsmiljö är tanken att man ska ha flera olika verktyg och leverantörer i en och samma miljö. I miljön delar verktygen på de gemensamma resurserna, till exempel information (via en så kallad repository eller resurskatalog). Om man lyckas åstadkomma en sådan miljö kan man byta och ändra i verktygen så att utvecklingsmiljön passar den egna organisationen. Problemet är att det inte finns några sådana miljöer framtagna idag, men det pågår arbeten på många håll. Arbetena grundar sig ofta på de erfarenheter man fått i utvecklingsmiljöer enligt punkt 1 och 2 ovan.

För att förstå vad en utvecklingsmiljö består av kan man se vad CASE-verktygen innehåller (se också figuren):

- *ett användargränssnitt* (specialskrivet eller standardiserat, t ex OSF/Motif, Microsoft Windows eller OS/2 Presentation Manager)
- en *databas med databasgränssnitt* (för att lagra och hantera informationen i verktyget)
- *export- och importfunktioner* (för att kunna flytta information till och från verktyget)
- *beroende funktioner* (dvs funktioner som måste finnas för att verktyget ska fungera)
- *oberoende funktioner* (dvs funktioner som finns i verktyget men som inte är nödvändiga för att verktyget ska fungera. Stödet för vissa metoder i verktyget är exempel på oberoende funktion)



Innehåll i CASE-verktyg

Om man strävar efter en utvecklingsmiljö som kan ändras, dvs som är öppen och oberoende av leverantörernas utveckling av verktygen, bör man försöka få öppenhet i alla de ovan beskrivna beståndsdelarna. Men det kan vara svårt och kanske inte ens eftersträvansvärt att göra verktygets beroende funktioner öppna och tillgängliga för förändring. Det finns undersökningar där man kommit fram till att endast 20-30% av verktygets funktioner är beroende funktioner och resterande 70-80% är oberoende funktioner. Kan man få öppenhet i de oberoende funktionerna har man uppnått stor öppenhet i verktyget och miljön.

Det finns i princip två sätt att skapa en sådan utvecklingsmiljö:

1. Man väljer en miljö där det som inte är verktygsberoende implementeras i miljön och sedan kompletteras med verktyg som fungerar i miljön. Exempel på sådana miljöer är IBM:s AD/Cycle, HP:s HP-Softbench och Digitals NAS med CDD/Repository. I de här miljöerna kallas verktygen C-CASE (Component CASE). Det här är den typ av utvecklingsmiljö som man bör sträva efter då den har hög flexibilitet och öppenhet. Problemet är att dessa miljöer inte är helt kompletta idag.
2. Det andra alternativet är att välja ett CASE-skal. I CASE-skal är det möjligt att ändra allt i CASE-verktyget utom de beroende funktionerna. Nackdelen i en sådan miljö är just att de generella delarna tillhör de beroende funktionerna i verktyget till skillnad från den miljö som beskrivs i ovanstående punkt. Fördelen är att den har kommit längre i utvecklingen och är möjlig idag.

2.2 Posten och Excelerator

I Posten har man under de senare åren skaffat sig ett antal hjälpmedel för att stödja systemutvecklingsprocessen, bl a finns det ett CASE-verktyg som heter DAX. DAX är en anpassning av CASE-verktyget Excelerator till metodiken SVEA/Direct, som är mycket vanlig i Posten. Den versionen av Excelerator kan användas i DOS-miljö. Excelerator i DOS-miljö har inte ändrats mycket sedan den lanserades i början av 1980-talet, men i mars 1992 kom en helt ny version av Excelerator, nämligen XLII. Den versionen används under operativsystemet OS/2 och har genomgått stora förändringar. Idag är den mer ett CASE-skal (se punkt 2 ovan) än ett CASE-verktyg.

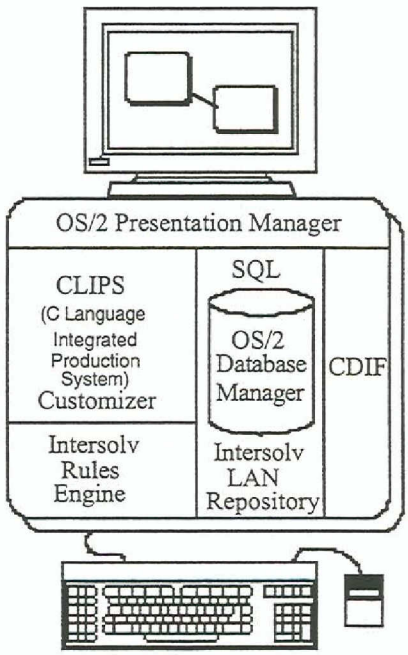
Eftersom Posten använder DAX, Excelerator under DOS, är det av intresse att se om XLII verkligen är den öppna produkt som beskrivits ovan. Men viktigast är att Postens DA-projekt håller på att ta fram en interimistisk resurskatalog – en databas som stödjer DA-verksamheten. Om XLII är det verktyg som det utger sig för att vara, kan det vara den resurskatalog som DA-verksamheten söker. Att ta reda på det är också huvudsyftet med denna analys av XLII:s öppenhet.

3 XLII-miljön

Intersolvs utvecklingsmiljö i XLII är ett CASE-skal. XLII-miljön består av följande komponenter (se också bilden):

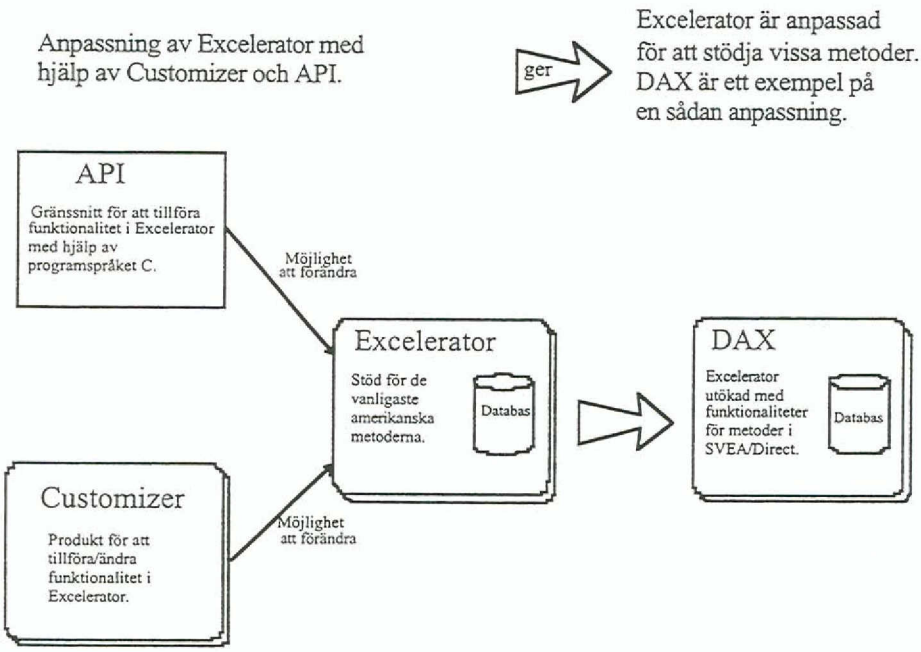
- *Användargränssnittet* är IBM:s OS/2 Presentation Manager.
- *Databasen* är Microsoft SQLServer eller IBM:s OS/2 Database Manager. I framtiden kommer databasen att kunna vara andra SQL-databaser eftersom man valt SQL som *gränssnitt* mot databasen.
- *Export och import* görs genom interimsstandarden CDIF, Case Data Interchange Format. CDIF är framtagen för att bli en standard, därför är den också lätt att använda för alla som vill exportera och importera information till XLII.
- För de *beroende funktionerna* används någonting som kallas IRE, Intersolv Role Engine, det är motorn som gör att man kan använda XLII-miljön.
- För att beskriva *oberoende funktioner* används en produkt som heter Customizer och ett språk som heter CLIPS (C Language Integrated Production System) som är framtaget av NASA i USA.

De viktigaste komponenterna i XLII-miljön är Intersolv Lan Repository (ILR), Intersolv Rules Engine (IRE) och Customizer som utgör grunden för miljöns öppenhet.

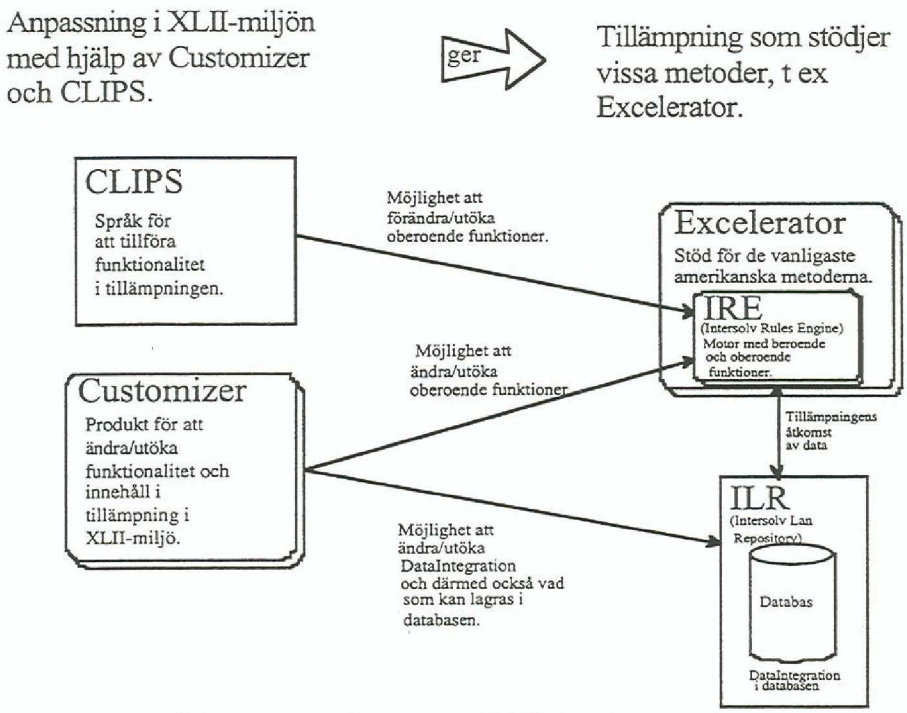


XLII-utvecklingsmiljö

Som framgår av bilden på föregående sida är stora delar av XLII en öppen miljö som är möjlig att förändra. Därför kan man klassificera XLII-miljön som ett CASE-skal. Skillnaden mellan DOS-versionen av Excelerator och XLII-miljön i OS/2 är att DOS-versionen gör det möjligt att *anpassa* Excelerator till kundens behov med hjälp av Customizer och ett programmeringsgränssnitt. I XLII-miljön däremot *tar man fram* Excelerator och andra produkter och tillämpningar med hjälp av Customizer och språket CLIPS. Excelerator för OS/2 är med andra ord bara en tillämpning framtagen med Customizer och CLIPS, IRE.



Anpassning av Excelerator i DOS med hjälp av Customizer och API



Tillämpningen Excelerator i OS/2 framtagen med hjälp av Customizer och CLIPS

3.1 Intersolv Lan Repository (ILR)

ILR är egentligen beskrivningen av innehållet i den databas där data lagras. Databashanteraren är inte framtagen av Intersolv utan man använder sig av SQL-databaser. Idag kan IBM:s Database Manager eller Microsoft SQLServer användas som databas. Meningen är att databasen ska vara placerad i ett lokalt nätverk (därav benämningen Intersolv *Lan* Repository), men den kan även installeras lokalt på en fristående persondator med OS/2. Beskrivningen av innehållet i ILR är hämtat från IBM:s arbete med att ta fram informationsmodeller för Repository Manager (RM). Informationsmodellerna har dock förändrats och utökats för att kunna användas i ILR. Modellerna bygger på Entity Relationship (ER). Modellerna har kompletterats med en hierarki av super- och subentitetstyper, där subentitetstypen ärver egenskaper från superentitetstypen. En entitetstyp är den typ av entitet som kan användas vid utvecklingsarbetet. Exempel på entitetstyp är Data Entity som gör att man kan använda entiteten Data Entity t ex i arbetet med att ta fram en datamodell.

3.2 Intersolv Rules Engine (IRE)

Den andra viktiga komponenten är motorn i utvecklingsmiljön – den beroende funktionen. Den kallas för Intersolv Rules Engine (IRE). IRE bygger på ett språk som heter CLIPS (C Language Integrated Production System) och har tagits fram av NASA i USA. De har även tagit fram en interpretator för CLIPS som Intersolv har utökat för att den ska passa som motor i utvecklingsmiljön.

3.3 Customizer

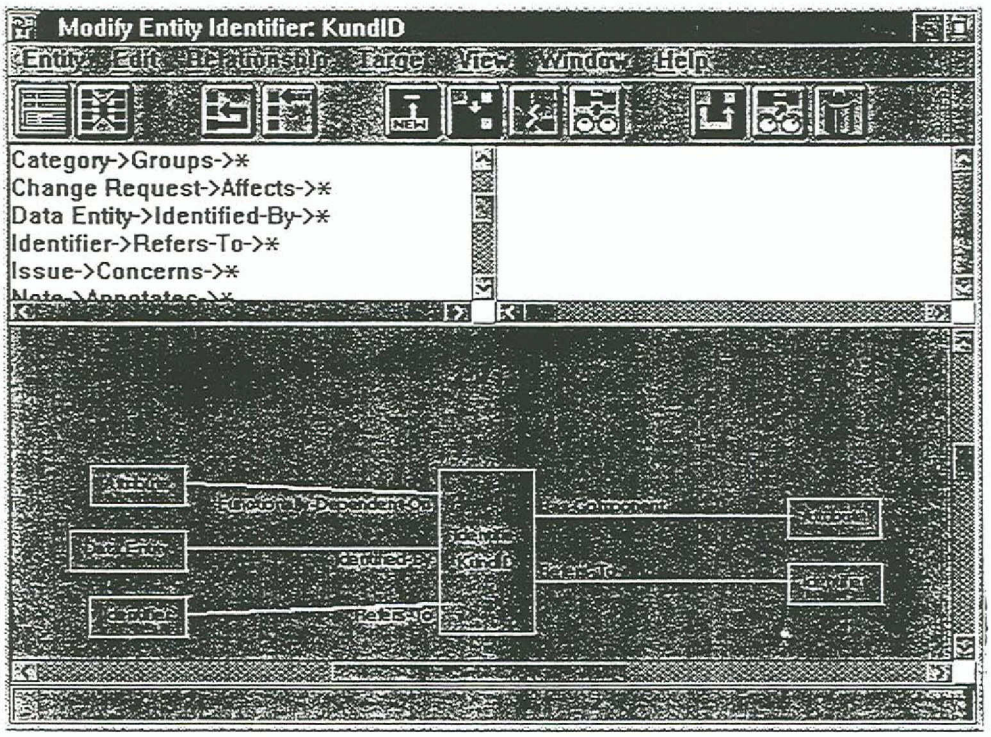
Customizer används för att utveckla tillämpningar som använder ILR och IRE. Den har t ex använts för att ta fram Excelerator – en tillämpning som kan användas i XLII-miljö. Den är alltså inte någon fristående produkt. Customizer tillsammans med CLIPS i IRE gör det möjligt att lägga till och förändra funktioner i IRE och entitetstyper i ILR. I princip går det att ta fram vilka tillämpningar som helst för att stödja utvecklingsprocessen. Den enda begränsningen är hur mycket arbete man vill lägga ned.

3.4 Övrigt intressant i XLII-miljön

Det finns några speciella faciliteter i miljön, förutom de ovan nämnda, som är lite speciella och som jag vill lyfta fram i detta sammanhang:

3.4.1 Repository Browser

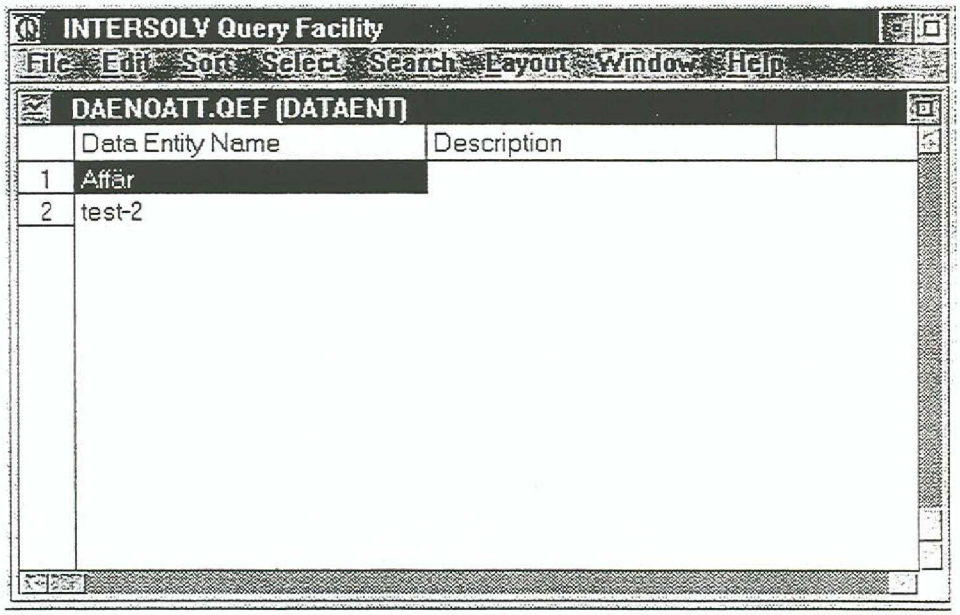
Repository Browser är en facilitet som man kan använda i tillämpningen för att se och behandla data i ILR (se figuren nedan). Den innehåller funktioner som gör att det går utmärkt att navigera i ILR.



Repository Browser

3.4.2 Query Facility

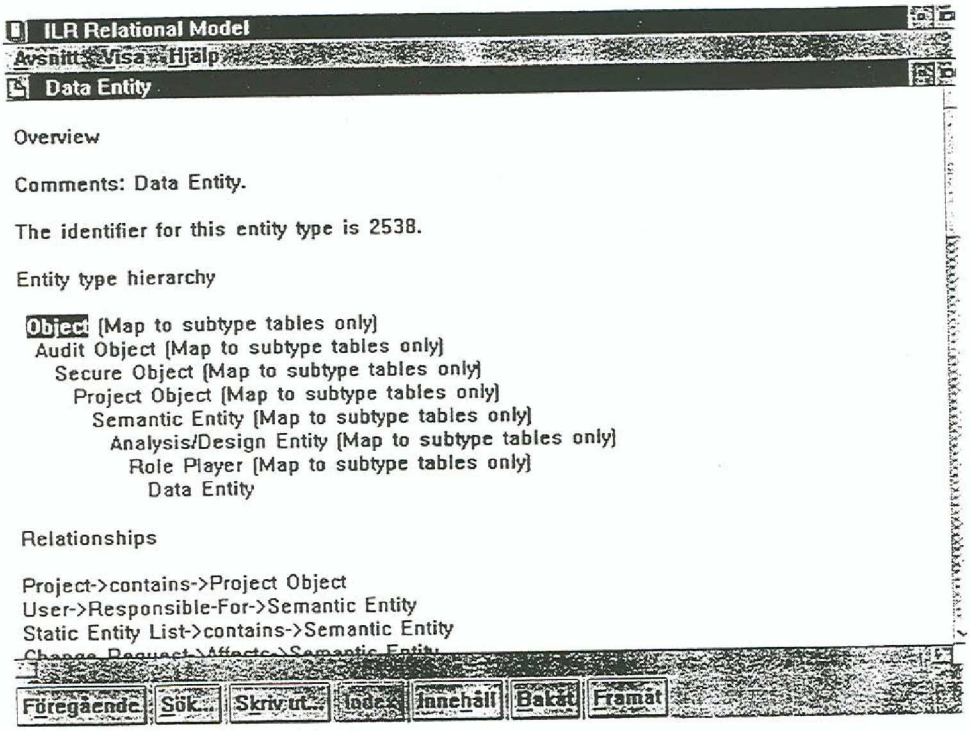
Query Facility gör det möjligt att på ett enkelt sätt att söka efter information i ILR som sedan kan användas bl a för att ta fram olika typer av rapporter (se figuren nedan). Resultaten kan också användas i andra hjälpmedel, till exempel ordbehandlare.



Query Facility

3.4.3 Structured Methodology Reference

Structured Methodology Reference är en facilitet som är fristående från tillämpningen. Den används för att visa hur informationen i ILR är uppbyggd med entitetstyper, relationer, egenskaper, entitetstypshierarki m m (se figuren nedan). Man kan där bland annat se var i entitetstypshierarkin en specifik entitetstyp är placerad och vilka egenskaper entitetstypen har ärvt från överliggande superentitetstyper.



Structured Methodology Reference

3.4.4 Case Data Interchange Format (CDIF)

För export och import till ILR har man inte tagit fram ett eget format utan man har valt CDIF (Case Data Interchange Format). CDIF är en interimsstandard för utbyte av data mellan CASE-verktyg. Även om det bara är en interimsstandard är det positivt att man valt att följa en öppen standard i stället för att hitta på ett eget format. Export och import till ILR genomförs med hjälp av IRE. Noteras bör att det också går att utföra export och import med IBM:s ESF-format.

4 Öppenhet och flexibilitet i XLII-miljön

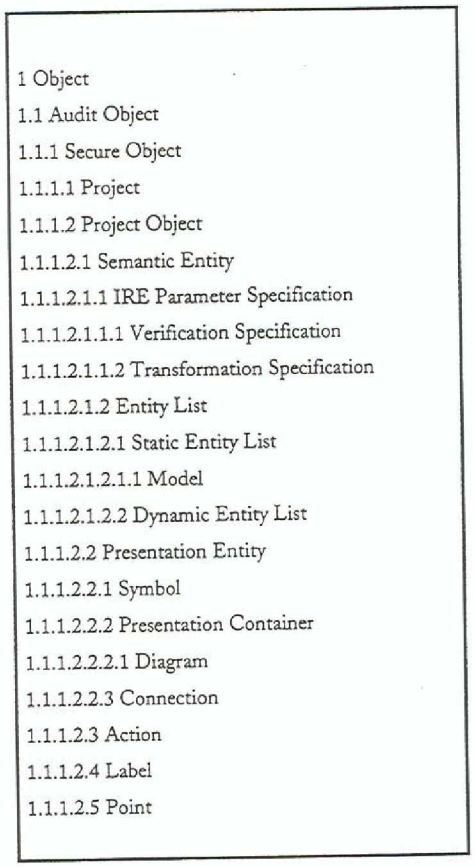
4.1 Val av databas

Som redan nämnts är den databas som används för att lagra data inte någon produkt från Intersolv. Den är i stället baserad på SQL-databaser. Idag går det att använda IBM:s Database Manager i OS/2 eller Microsoft SQLServer. I Intersolvs planer ligger att man i framtiden ska kunna använda ett större antal SQL-databaser. Det gör det möjligt att välja den SQL-databas som passar bäst.

4.2 Förändring och utökning av innehållet i resurskatalogen

Innehållet i databasen, som ingår i dataintegrationen (DI), grundar sig på delar av de informationsmodeller som tagits fram av IBM för Repository Manager. Intersolv har i vissa delar kompletterat dessa modeller. DI innebär att verktygen i miljön förstår och hanterar data på samma sätt. För att kunna göra det måste DI vara tillgängligt för alla som vill göra tillämpningar till ILR.

Med hjälp av Customizer är det möjligt att göra förändringar eller utöka innehållet i ILR. Det går att förändra allt utom det som tillhör den fundamentala submodellen som är grunden för ILR. Bilden på nästa sida beskriver den fundamentala submodellen och visar entitetstyperna i hierarkisk form där man ser super- och subentitetstyper. En subentitetstyp ärver egenskaper från superentitetstypen; Secure Object ärver till exempel från Audit Object.

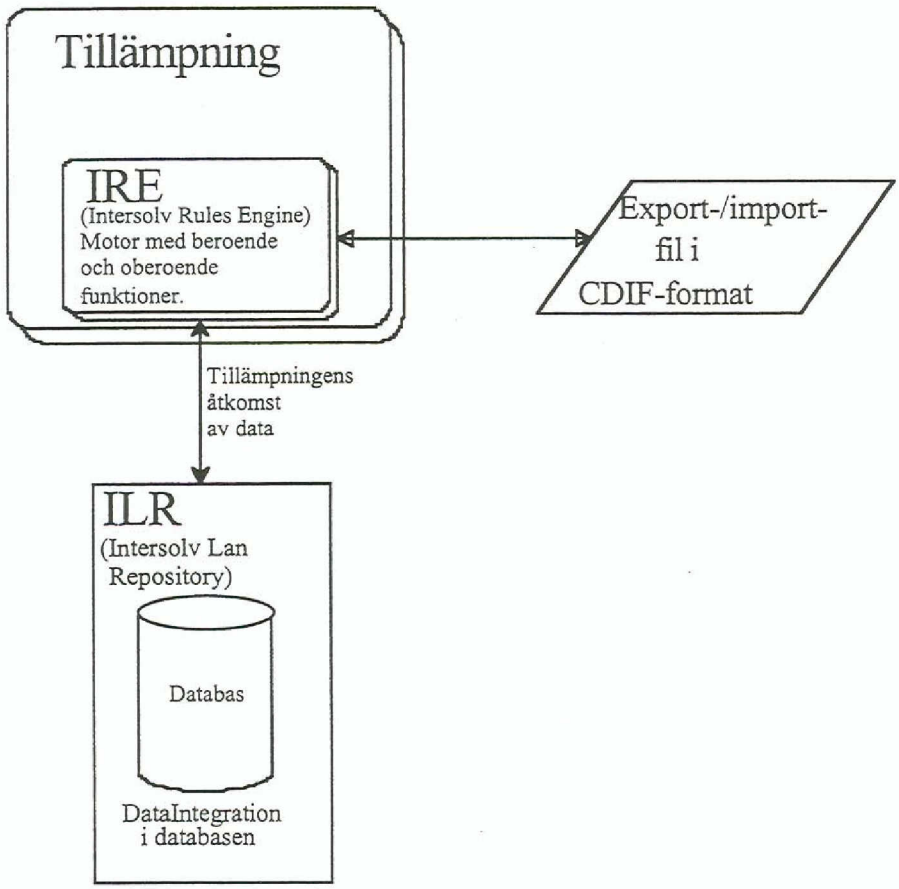


Fundamental submodell (hierarkisk form)

Man kan förändra och utöka ILR med entitetstyper, egenskaper och relationer. Entitetstyperna är placerade i en hierarkisk struktur med super- och subentitetstyper. En subentitetstyp ärver egenskaper från de överliggande superentitetstyperna. Det gör att det är lätt att införa generella egenskaper, t ex säkerhet, till nya entitetstyper genom att placera dem under en viss superentitetstyp.

4.3 Export och import

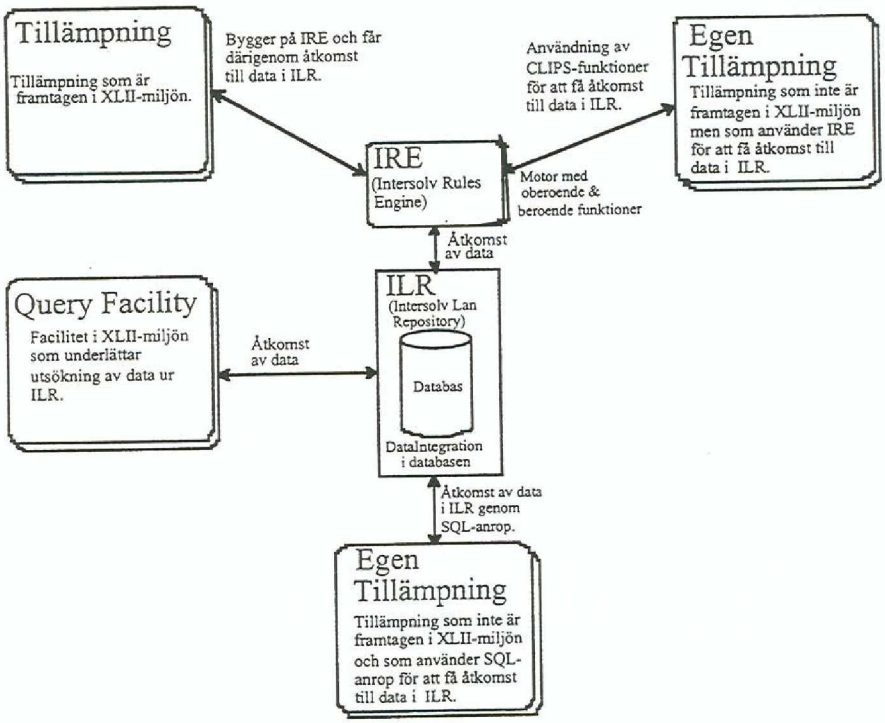
För att exportera eller importera data till ILR används CDIF (Case Data Interchange Format). I CDIF har man skilt på syntax, dvs utseendet på det som överförs, och semantik, dvs vad det är som ska överföras. När man ändrar eller utökar ILR ändras eller utökas också det som kan exporteras eller importeras till ILR utan att syntaxen behöver ändras. Det är positivt att formatet är en standard som gör det möjligt för andra verktygstillverkare att använda formatet vid export och import till ILR. Det går också att använda IBM:s format ESF för export och import. Det formatet är inte lika innehållsmässigt som CDIF men det används av flera LowerCase-verktyg.



Export och import till ILR genom CDIF

4.4 Åtkomst till resurskatalogen ILR

I normala fall sker åtkomsten av data i ILR genom IRE, men tack vare att databasen är en neutral SQL-databas är det möjligt att komma åt data i ILR med SQL-kommandon eller ta fram tillämpningar med SQL-anrop.



Åtkomst till data i ILR

4.5 Förändring och utökning av funktioner

Genom att använda Customizer och språket CLIPS är det möjligt att förändra och utöka funktionaliteten i en tillämpning för att till exempel stödja en viss metod eller för att ta fram en egen tillämpning. Det går också att lägga till egna analyser. En analys kan vara av två olika typer:

1. automatisk (trigger), dvs kontrollen sker under tiden man utför ett arbete med exempelvis datamodeller
2. användarstyrd, dvs analysen startas av användaren vid en viss tidpunkt

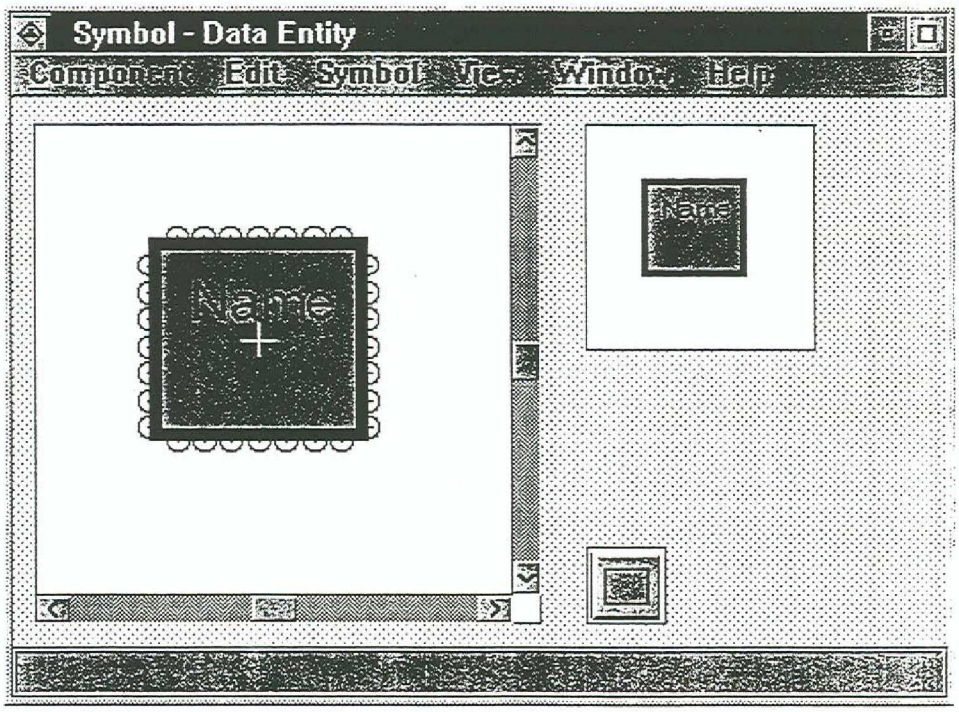
5 Customizer

Produkten Customizer är en viktig komponent för flexibiliteten och öppenheten i XLII-miljön. Med Customizer kan det mesta i miljön förändras. Nedan är de flesta och viktigaste komponenterna översiktligt beskrivna.

5.1 Symbol

Det är möjligt att göra ändringar i och lägga till symboler som används i diagrammen. Det går att

- ändra och skapa utseende för symboler (se figuren nedan)
- definiera egenskaper för symboler
- bestämma åtgärder (actions), dvs koppla funktioner som kan utföras av användaren till symbolen
- skriva led- och hjälptexter (prompter), som ska visas tillsammans med symbolen
- skapa bindningar till den entitetstyp eller relation i ILR som symbolen representerar (en entitetstyp eller relation i ILR kan ha flera olika symboler)



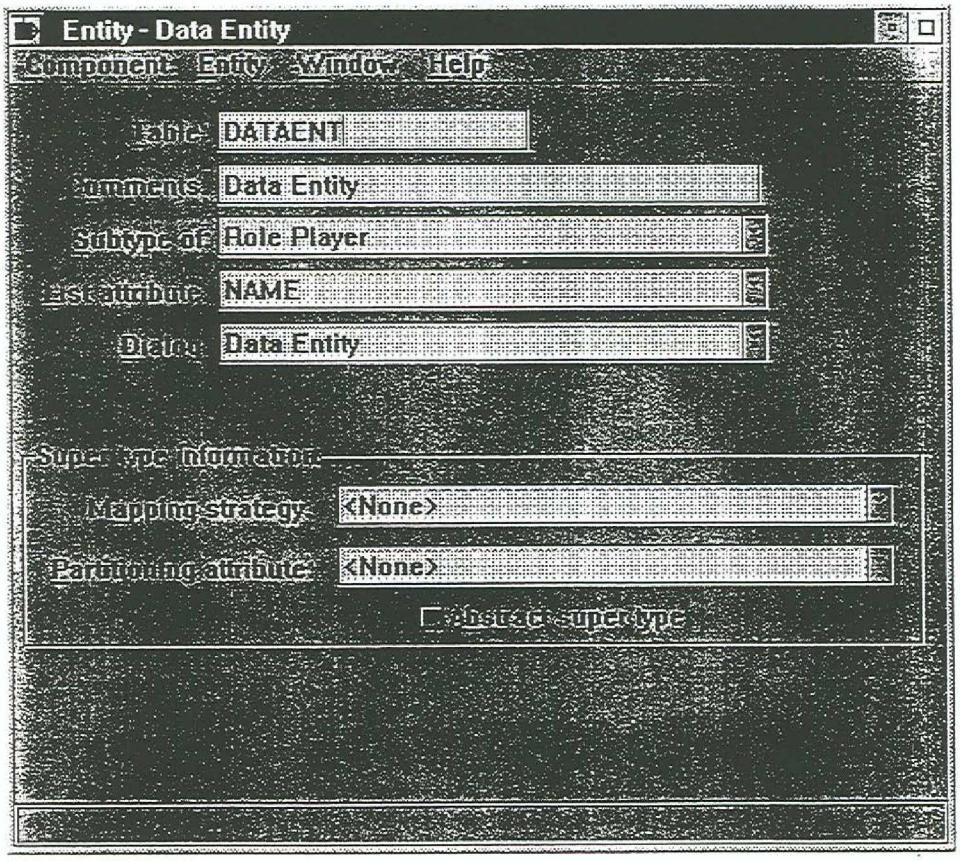
Symbol för Data Entity

5.2 Entitetstyp, egenskap och relation

Det är möjligt att lägga till, ändra och ta bort entitetstyper, egenskaper och relationer i ILR. Man anger också vilken entitetstyp som är entitetstypens superentitetstyp, dvs var i arvshierarkin entitetstypen ska placeras för att ärva de överliggande entitetstypernas egenskaper (i figuren nedan "Subtype of:").

Dessutom är det möjligt att göra följande:

- ange vilken dialog som ska användas till entitetstypen ("Dialog:" i figuren nedan)
- ange egenskaper för entitetstyper och relationer
- beskriva egenskaper och relationer mellan entitetstyper



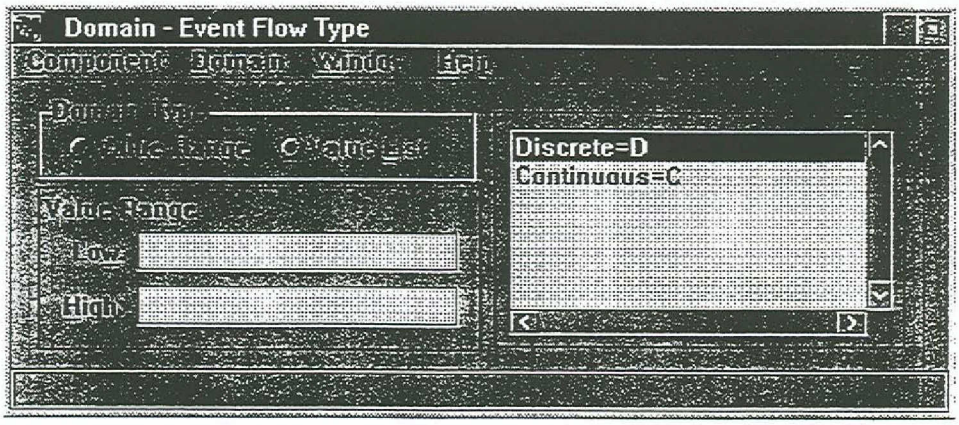
Entitetstypen Data Entity

5.3 Domän

Möjlighet finns att definiera värdemängder som ska gälla för bl a entitetstyper och egenskaper.

Värdemängderna kan vara något av följande:

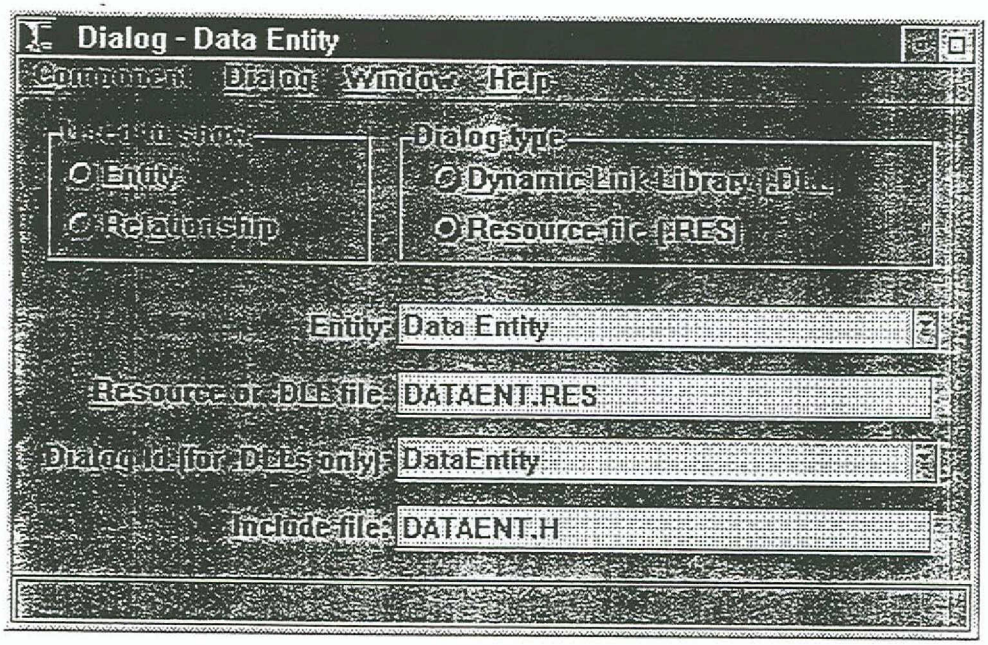
- ett värdeområde ("Value Range" i figuren nedan)
- en lista av olika värden ("Value List" i figuren nedan)



Domänen Event Flow Type

5.4 Dialog

Man kan definiera dialoger för entitetstyper och relationer. I beskrivningen anges vilken fil som innehåller dialogen. Dialogen måste idag skapas i OS/2 Toolkit Dialog Box Editor, som ingår i IBM:s utvecklingsverktyg för OS/2 och som ligger utanför XLII-miljön. Det här upplever jag som en begränsning i vad som kan göras med Customizer. I framtiden borde den funktionen ingå i Customizer och använda den grafiska bildskärmsmålaren GUI som ingår i XLII-miljön.

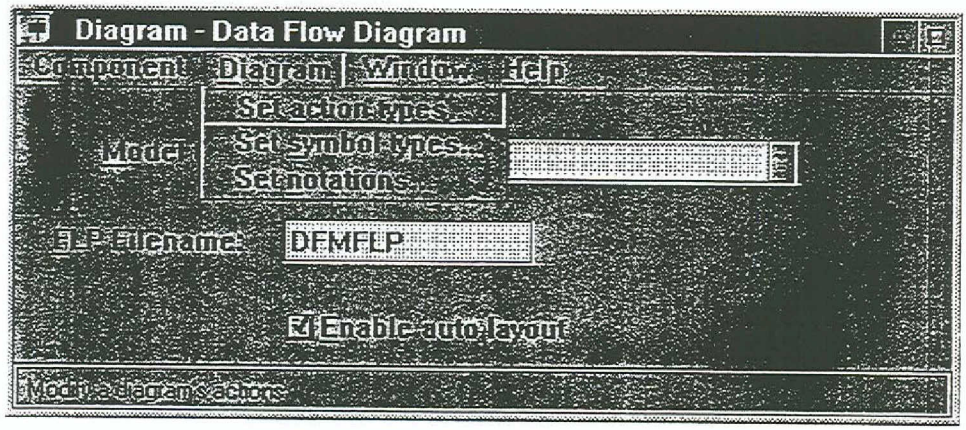


Beskrivning av dialogen för entitetstypen Data Entity

5.5 Diagram

Det är möjligt att specificera de diagram som kan användas. I specifikationen anger man bland annat:

- vilka åtgärder (se "Set action types..." i figuren nedan) som är möjliga att göra för diagrammet – det är till exempel möjligt att ha en åtgärd som överför en logisk datamodell till en fysisk datamodell enligt en viss princip.
- vilka symboler (se "Set symbol types..." i figuren nedan) som ska kunna användas i diagrammet.
- vilken notation (se "Set notations..." i figuren nedan) som diagrammet följer, till exempel Merise.



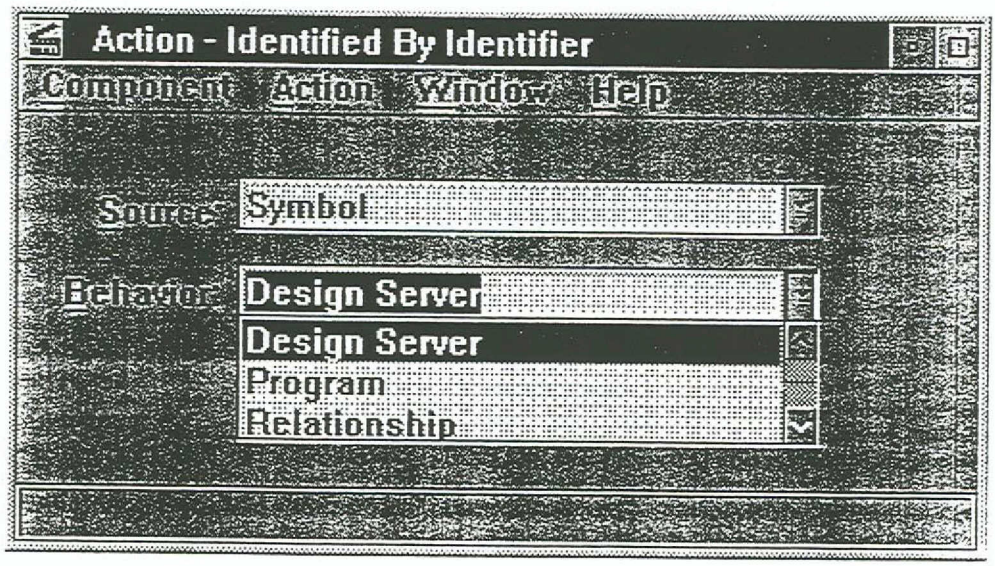
Beskrivning av diagrammet Data Flow Diagram

5.6 Åtgärder (Actions)

Möjligheterna att skapa åtgärder (actions), beskrivs i figuren nedan. Åtgärder är funktioner utöver de som man automatiskt får med i miljön. Exempel:

- åtgärder som ger stöd för en viss metod, t ex analys som är användarstyrd eller automatisk
- åtgärder som inte bygger på en viss metod utan i stället är tekniska lösningar, t ex åtgärder som ger möjlighet att starta andra tillämpningar

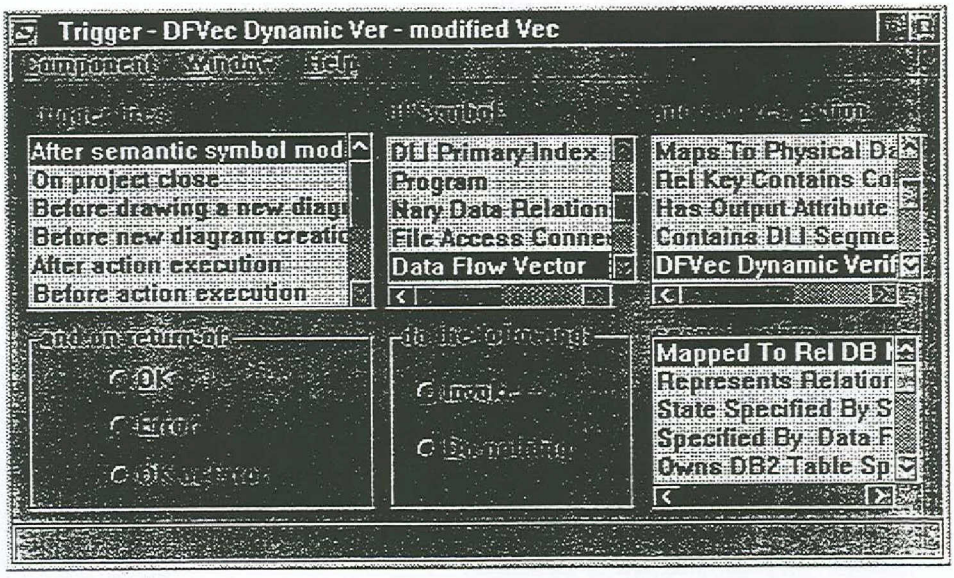
Här kan man bl a koppla egna åtgärder, funktioner, till det som ska skrivas i CLIPS. Det åstadkoms genom att "Design Server" sätts i boxen "Behavior:", se figuren nedan.



Åtgärd (action) Identified By Identifier

5.7 Händelser (Triggers)

Man kan också bestämma de händelser som automatiskt utlöser en viss åtgärd (action). Den här faciliteten är speciellt intressant när det gäller att automatiskt kunna kontrollera om det går att förbinda olika symboler med varandra (se figuren nedan).

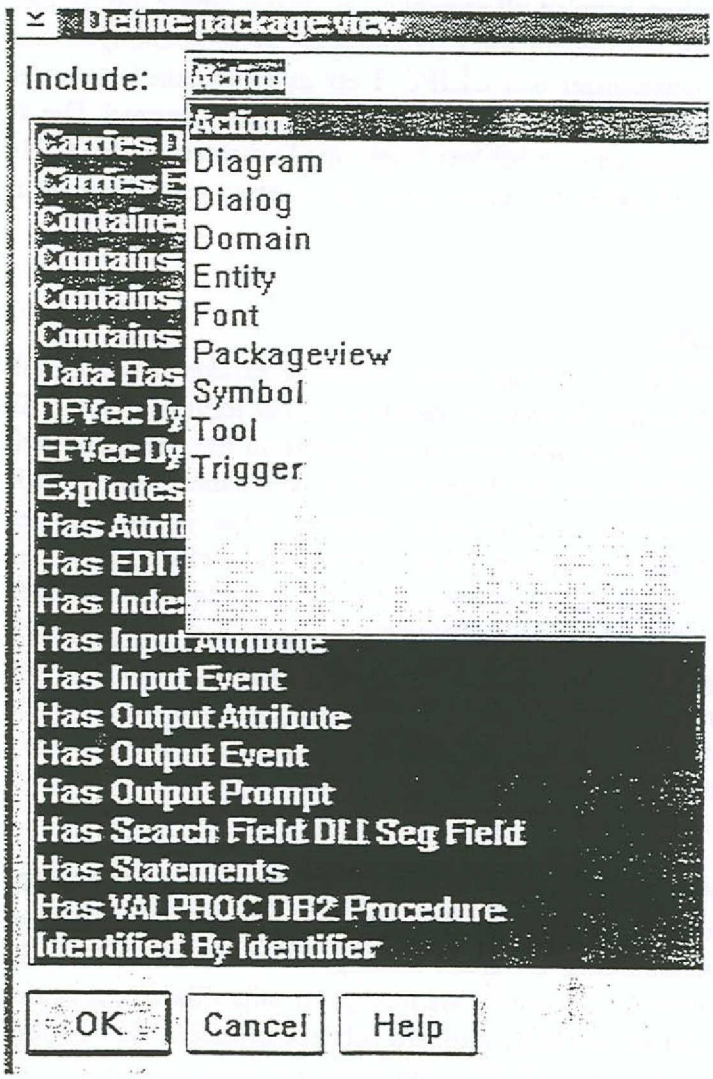


Händelse (trigger) DFVec Dynamic Ver – modified Vec

5.8 Tillämpning (Package)

Till sist går det att specificera vad som ska ingå i en tillämpning (package). Det innebär att man med hjälp av Customizer kan sätta ihop olika tillämpningar utifrån en grundstomme. För varje tillämpning bestämmer man bl a vad som ska ingå av följande:

- symboler
- entitetstyper
- domäner
- dialoger
- diagram
- åtgärder (actions)
- händelser (triggers)



6 Begränsningar i XLII-miljön

Som framgått av föregående avsnitt är XLII-miljön mycket öppen och flexibel. Men det finns tre stora begränsningar som jag vill redogöra för i det här avsnittet.

6.1 Brist på mognad eller kunskap

Den första begränsningen är XLII-miljöns beroende av den kunskap eller mognad som finns i den eller de organisationer där den ska implementeras. Öppenheten och flexibiliteten i miljön innebär i sin tur att den blir komplex att hantera både när det gäller innehåll i ILR och funktioner i IRE. Komplexiteten är inte enbart kopplad till utveckling av tillämpningar som stödjer systemutvecklingen utan också till användningen av tillämpningar som tagits fram med Customizer och CLIPS. I ett inledningsskede kan det krävas extra resurser för att bygga upp denna kunskap eller mognad. Det är egentligen inte konstigt att man behöver bygga upp kompetensen om XLII-miljön – samma sak gäller för andra typer av verktyg och miljöer som används idag.

6.2 Långsam

Den version av XLII som har använts i studien är mycket långsam. Till försvar för produkten kan nämnas att maskinvaran har legat på gränsen till vad som är behövligt för implementationen. Dessutom har databasen varit lokalt installerad vilket också kan bidra till att databashanteraren har tagit upp lokala systemresurser som belastar systemet. Men jag har sett en installation med rejält tilltagen lokal systemkonfiguration och med databasen installerad på en annan dator. Den var åtskilligt snabbare men fortfarande inte tillräckligt snabb för att kunna användas i en produktiv systemutvecklingsmiljö. Om Intersolv kan lösa detta problem eller om maskinvaran blir kraftfullare, är det min fulla övertygelse att det här är en milstolpe för integrerade utvecklingsmiljöer och framför allt för de som använder datorstöd i utvecklingsarbetet.

6.3 Antalet SQL-databaser som kan användas

En annan begränsning som finns i miljön idag är att det bara är möjligt att använda SQL-databaserna Database Manager i OS/2 och Microsoft SQLServer som databas. Det borde vara möjligt att välja vilken databas som helst bara den följer en viss SQL-standard och kan installeras på ett lokalt nätverk. Om den sedan körs i UNIX-miljö skulle inte spela någon roll.

7 Sammanfattning och råd

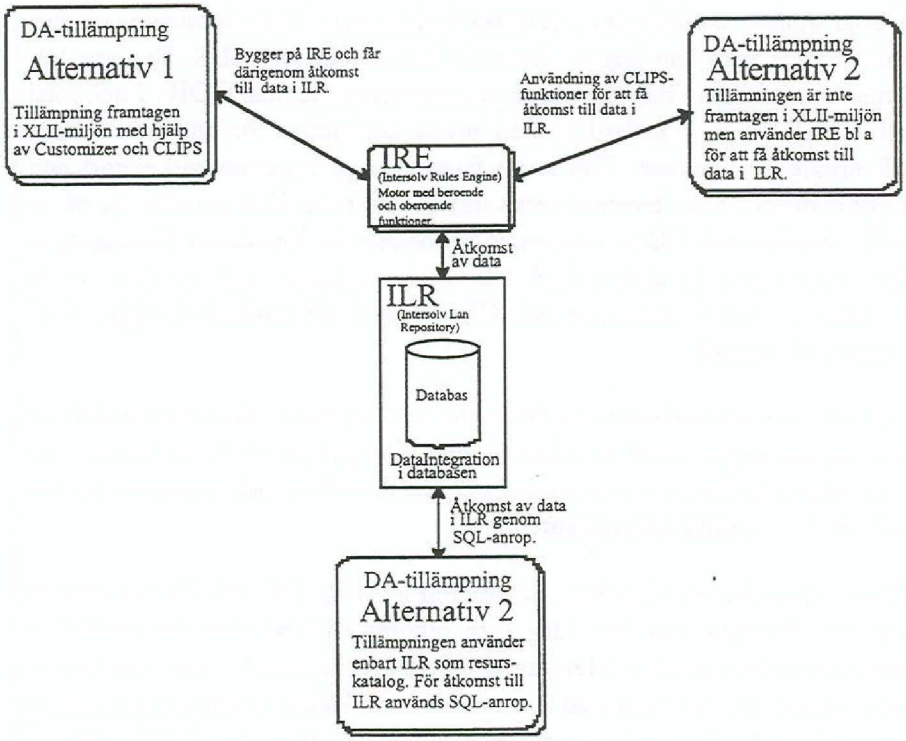
I Intersolvs XLII-miljö får man en miljö som är mycket öppen och flexibel. Det är möjligt att förändra och utöka funktionerna i tillämpningen och den information som kan lagras i Intersolv Lan Repository, ILR. Export- och importfunktionen i miljön baseras på interimsstandarden, CDIF. I och med att CDIF är en standard är den också användbar för andra verktygstillverkare än Intersolv. Det kan i framtiden ge öppenhet vid export och import mellan olika leverantörers verktyg. Eftersom ILR grundar sig på en SQL-databas och IBM:s informationsmodeller för Repository Manager går det att komma åt data i ILR från andra tillämpningar än de som är framtagna med Customizer och CLIPS utan att använda bryggor med export och import.

Allt det ovan nämnda ger stor flexibilitet och öppenhet. Men hög flexibilitet och öppenhet ger också stor komplexitet. Det gör att det krävs kompetens och planering för att kunna utnyttja denna flexibilitet och öppenhet på rätt sätt så att inte miljön blir ohanterlig.

Den testversion som använts vid studien av ILR, IRE och Customizer är mycket långsam och kan idag inte rekommenderas som en produktiv systemutvecklingsmiljö. Men jag är mycket imponerad av hur man byggt upp miljön och är övertygad om att om man lyckas få bättre prestanda i miljön kommer Intersolvs produkter att inleda en ny era för CASE-miljöer, nämligen CASE-skal. För DA:s behov anser jag att den mycket väl kan tjänstgöra som både interimistisk resurskatalog och tillämpning.

Jag rekommenderar först och främst att man gör en tillämpning för DA med hjälp av Customizer, IRE och CLIPS, och använder ILR som interimistisk resurskatalog. ILR bör installeras på ett nätverk som är åtkomligt för alla DA-medlemmar. Resurskatalogen, ILR, håller tillräcklig kvalitet för att även kunna bli DA:s permanenta resurskatalog. Men om den bara ska tjäna som interimistisk katalog bör det vara lätt att flytta informationen till en annan resurskatalog. Det är också möjligt att integrera DA-funktioner, systemutvecklingsfunktioner och systemutvecklingsinformation i miljön. Det skulle underlätta DA:s roll vid systemutveckling. Dessutom får man många andra synergieffekter, bl a i form av den Repository Browser som finns i miljön och som underlättar åtkomst och behandling av data i resurskatalogen. Den färdiga tillämpningen kan sedan genereras. Tillämpningen används då utan Customizer medan IRE och ILR fortfarande behövs.

Om det inte är aktuellt att ta fram denna tillämpning vill jag som ett andra alternativ rekommendera att man använder ILR som resurskatalog och hämtar och bearbetar data i ILR genom att använda IRE eller CLIPS, eller genom direkta SQL-anrop. Men man måste beakta att om man behöver utöka resurskatalogen, ILR, med speciell DA-information är det bäst att använda Customizer.



DA-tillämpning enligt alternativ 1 och 2

TIDIGARE UTGIVNA PUBLIKATIONER AV TRIADGRUPPEN

Verksamhetskrav på informationsadministration

V 1: IA och verksamhetens krav – erfarenheter från offentlig förvaltning
V 2: Fallstudie av IA-projektet vid Televerket
V 3: IA-erfarenheter från företag och myndigheter
V 4: Den gemensamma informationsmarknaden – en referensram för handlingsfrihet och konkurrenskraft

Modellering

N 1: Modelleringsansatser för begrepps- och datamodellering – Beskrivning och försök till jämförelse
N 2: Generering av konceptuella modeller från policydokument
N 3: Espritprojektet Tempora
N 4: Prövning av regelbaserad metodik inom Posten
N 5: En kokbok i remodellering – utkast
N 6: Datorstöd för modellintegration
N 7: Modellbaserad kunskapsinsamling
N 8: Modellkvalitet
N 9: Samband mellan dokument och modeller

Kunskapsförmedling

H 1: Handledarutbildning för modelleringsledare, avancerad
H 2: Slutrapport HUMLA prototyp
H 3: Utbildning i Informationsadministration

Uttagssystem

U 1: Hybris i Unix-miljö
U 2: DEBRIS
U 4: Program för sökning i databaser – en marknadsöversikt
U 5: Att nå och förstå data – möjligheter och begränsningar

Katalogprinciper

K 1: IRDS
K 2: IRDS Modeller och modellnivåer
K 3: Kopping begreppsmodell – relationsmodell
K 4: IBM:s Repository Manager – en Introduktion
K 5: IBM:s Repository Manager: Datamodelleringsbegreppen
K 6: IBM:s Repository Manager: Begreppsmodellering i Information Model
K 7: IBM Repository Manager: Attribut- och värdemodellering i Enterprise Submodel
K 8: Navigering i Repository
K 9: TRIAD Newsletter – IRDS inom ISO. Dagsläget
K 10: TRIAD Newsletter – ISO/IRDS. Händelseutvecklingen 91/92
K 11: Samverkan mellan resurskataloger – visioner eller behov
K 12: AD/Cycle l Information Model – Processer och och informationsflöden mellan processer
K 13: AD/Cycle l Information Model – Info Flows inom Processmodellen
K 14: AD/Cycle l Information Model – Relationsdatabasmodellering
K 15: AD/Cycle l Information Model – Härledningsspecifikationer i begreppsmodellen
K 16: IA-prototyp
K 17: Repository AD/Cycle – International Users Group
K 18: RAD-konferensen i Chicago, 1992
K 19: Vad händer inom ANSI-IRDS?
K 20: Information Warehouse – vad är det?
K 21: CDIF – en översikt
K 22: PCTE – en översikt

KORT OM TRIAD

Triad är namnet på ett treårigt samarbetsprojekt kring informations-administration och dataadministration, IA/DA, som Televerket, Posten, Ericsson, Statskontoret och SISU bedriver. Syftet är att utveckla parternas synsätt, metoder och hjälpmedel inom detta område. Arbetet inom Triad är uppdelat i delprojekt som är sammanförda i tre block.

Beställarblocket vänder sig dels till dem som är verksamhetsansvariga och måste ta ställning till IA/DA-satsningar, dels till dem som har ansvaret för IA/DA inom en organisation. Delprojekten inom detta block arbetar med att formulera verksamhetens krav på IA/DA samt studerar och beskriver roller, organisation och arbetsformer för IA/DA-arbete.

Utförarblocket vänder sig till dem som arbetar med IA/DA. Delprojekten arbetar med modellering, data- och resurskataloger samt uttagssystem.

Kunskapsförmedling är det block som ser till att resultaten kommer Triad-parterna till godo. Detta sker bland annat genom kurser, seminarier samt genom att rapporter, som denna, ges ut.