TRIAD

Newsletter K 30

Objektorienterade ansatser inom ANSI/IRDS

Stig Berild

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

Innehåll

Objektorienterade ansatser inom ANSI/IRDS

I artiklar, på konferenser och i standardiseringssammanhang hänvisas till att IRDS-standarderna i och med anpassning till objektorienterad uppbyggnad och objektorienterade begrepp kommer att komma i fas med såväl andra standardiseringsaktiviteter som andra modernt uppbyggda system. Därmed kommer dessa nya standarder att anammas i en utsträckning som aldrig blev fallet med tidigare versioner. Föreliggande newsletter försöker reda ut hur långt arbetet kommit.

Läsaren förutsätts ha tagit del av tidigare utgivna IRDS-inriktade rapporter inom TRIAD-projektets område Katalogprinciper.

1 Bakgrund

IRDS-standarder har dels tagits fram inom ISO, dels inom ANSI (se Triad-rapporter K1, K2, K9, K10, K19). ISO använder sig av relationsmodellens begreppsapparat för modellering av IRD-data och IRD-scheman. ANSI hämtar begreppen från en variant av en Entity-Relationship modell. I båda fallen formulerar man ett gränssnitt mot ett IRDS i form av ett antal instruktioner definierade i termer av respektive modell. I ett övergripande perspektiv är ansatserna mycket snarlika, skillnaden ligger främst i begreppsapparaten. Kännetecknande är synen på ett IRDS som ett avancerat men för övrigt konventionellt dbms, d v s där data är passivt och hanterat av en generell mekanism i form av ett antal funktioner (services eller information management processes). Detta perspektiv vill nu ANSI "modernisera" genom ett mer objektorienterat synsätt. Statiska tupler och entities ersätts med objekt representerande både tillstånd och beteenden.

Föreliggande rapport beskriver aktuella ansatser så som de formuleras i ANSI-dokumentet X3H4/93-048R1 "Information Resource Dictionary System (IRDS), Services Architecture, Technical Report", juni 1993 och Robert Hodges: "The move of IRDS repository standards toward object orientation", Computer Standards & Interfaces nr 15.

Observera att det är fråga om en Technical Report och en artikel, inte en standard. En Technical Report kan innehålla utredningar, analyser, skisser, idéer. Rapporten är sannolikt första steget i en lång utvecklingsprocess mot IRDS-standarder baserade på ett mer objektorienterat angreppssätt. Noteras bör att rapporten ännu befinner sig i en "preliminary" utgåva.

Inom parentes passar jag på och listar några pågående standardiseringsarbeten inom ISOs och ANSIs IRDS-kommittéer.

Inom ISO:

- *IRDS 1.5 Services Interface Extensions to IS10728.*

  Modifieringar av IS10728. Inga revolutionerande nyheter. Antagligen DIS till sommaren.

- *IRDS Framework Standard, Revision.*

  Revidering, modernisering av IS10027. Är i WD-skick. Börjar få viss influens av objektorientering, men grundtankarna från IS10027 finns kvar.

Inom ANSI:

- *IRDS Repository Context Reference Model, X3/TR-12-93.*

  Technical Report. Skissar en generell indelning av repositoryfunktionalitet och viktiga gränssnitt.

- *IRDS Conceptual Schema, Part 1: Conceptual Schema for IRDS. X3H4/93-196, Part 2: Modeling Language Analysis.*

  Part 1 formulerar ett ramverk för representation, integrering och översättning av och mellan olika fristående representationsspråk och scheman. Part 2 redovisar ett bakgrundsmaterial i form av en genomgång av olika existerande modelleringsspråk.

- *IRDS Normative Schema.*

  ANSIs benämning på det som i IS10027 kallas IRD Definition Schema Level. Logikbaserat. Conceptual Graphs som språk diskuteras. Ska bli ANSI-standard.

Det intressanta med närmandet till objektorienterad specifikation är att det innebär en påtaglig modernisering av det sätt på vilket IRDS-problematiken attackeras. Samtidigt kan det finnas en hel del fallgropar och skapas förväntningar som kanske inte helt kommer att kunna infrias. Orsaken kan till exempel vara att ett IRDS har en karakteristik som inte till alla delar naturligt låter sig inordnas som en typisk OO-modell. Framtiden får utvisa bärkraften kring idéerna. I avsnitten 2-3 förklaras ansatsen. Avsnitt 4 innehåller avslutningsvis några subjektiva synpunkter.

2 Generell idéskiss

Det primära syftet med ett IRDS är att hantera beskrivning av en verksamhet och dess informationssystem, såväl dess dynamiska som dess statiska egenskaper. Ofta begränsar man intresset till informationssystemens livscykel, d v s ända från analys och design över realisering till drift och avveckling. Oavsett var man sätter gränserna, gäller grovt att data hanteras med hjälp av en uppsättning servicefunktioner. Se figur 1.

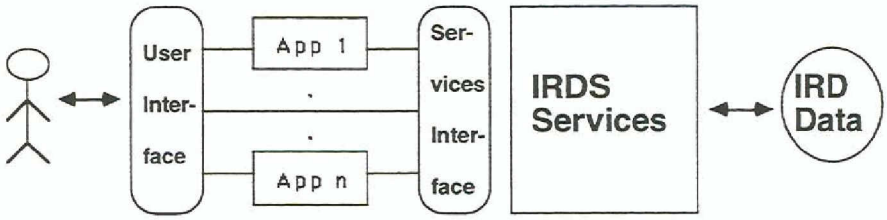
Figur 1

Dessa IRDS Services är i första hand avsedda att ge ett generellt stöd för de typer av hanteringar som bedöms erforderliga för tillämpningsområdet (IRD-hantering) ifråga. Man kan se ett IRDS som ett databashanteringssystem speciellt anpassat till de krav som tillämpningsområdet ställer. Aktuella data (IRD Data) kan förväntas vara av många olika slag och behöva hanteras på många olika sätt samt presenteras för många olika kategorier användare. Vi preciserar bilden något. Se figur 2.

I det allmänna fallet önskar nog användaren få data bearbetat, tillrättalagt och sammanställt på ett för behovet smakligt sätt. Det arbetet utförs av specifika tillämpningar. (*Applications*). I en utbyggd IRDS-miljö kan antalet tillämpningar förväntas bli omfattande. De kommer dessutom med största sannolikhet att samverka på olika sätt.

Samordning och flexibilitet underlättas om alla dessa tillämpningar kan formulera sina behov mot repositoryt på samma sätt genom ett standardiserat gränssnitt, en standardiserad uppsättning operationer med tillhörande syntax (*Services interface*). På så vis kan tillämpningar lättare tillföras och bytas ut med hänsyn till förändrade behov, konkurrens m m.

Därtill behövs ett enhetligt användargränssnitt eftersom användare måste kunna känna igen sig och vara produktiva oavsett vilken tillämpning de för tillfället jobbar mot (*User interface*).



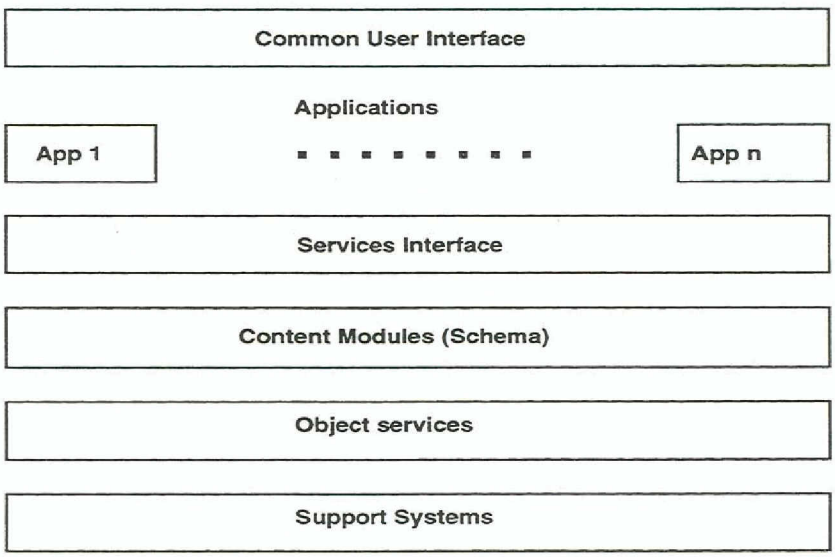
Figur 2

En ännu mer uppdelad struktur visas i figur 3 och motiveras enligt följande.

Även det man pratar om i Services Interface kan behöva standardiseras till sin betydelse (semantik). Detta är viktigt eftersom olika tillämpningar och olika användare kan förväntas arbeta mot delvis samma komponenter i repositoryt (*Content modules*).

Content modules behöver formuleras i något överenskommet språk eller begreppsapparat (meta meta model) för att olika användare eller tillämpningar ska vara säkra på att inte tolka in olika saker hos de definierade komponenterna. Detta språk präglar ju i högsta grad hur content modules byggs upp och vad de kan uttrycka. IRDS Conceptual Schema från dokumentet X3H4/93-196 förmodas bli använd som begreppsapparat för ändamålet ifråga.

IRDS-miljön bör om möjligt byggas upp med hjälp av existerande komponenter av olika slag. Dit kan höra ett stödjande databashanteringssystem, kommunikationsmekanismer för en distribuerad miljö, behörighetsmekanismer m m (*Support services*).



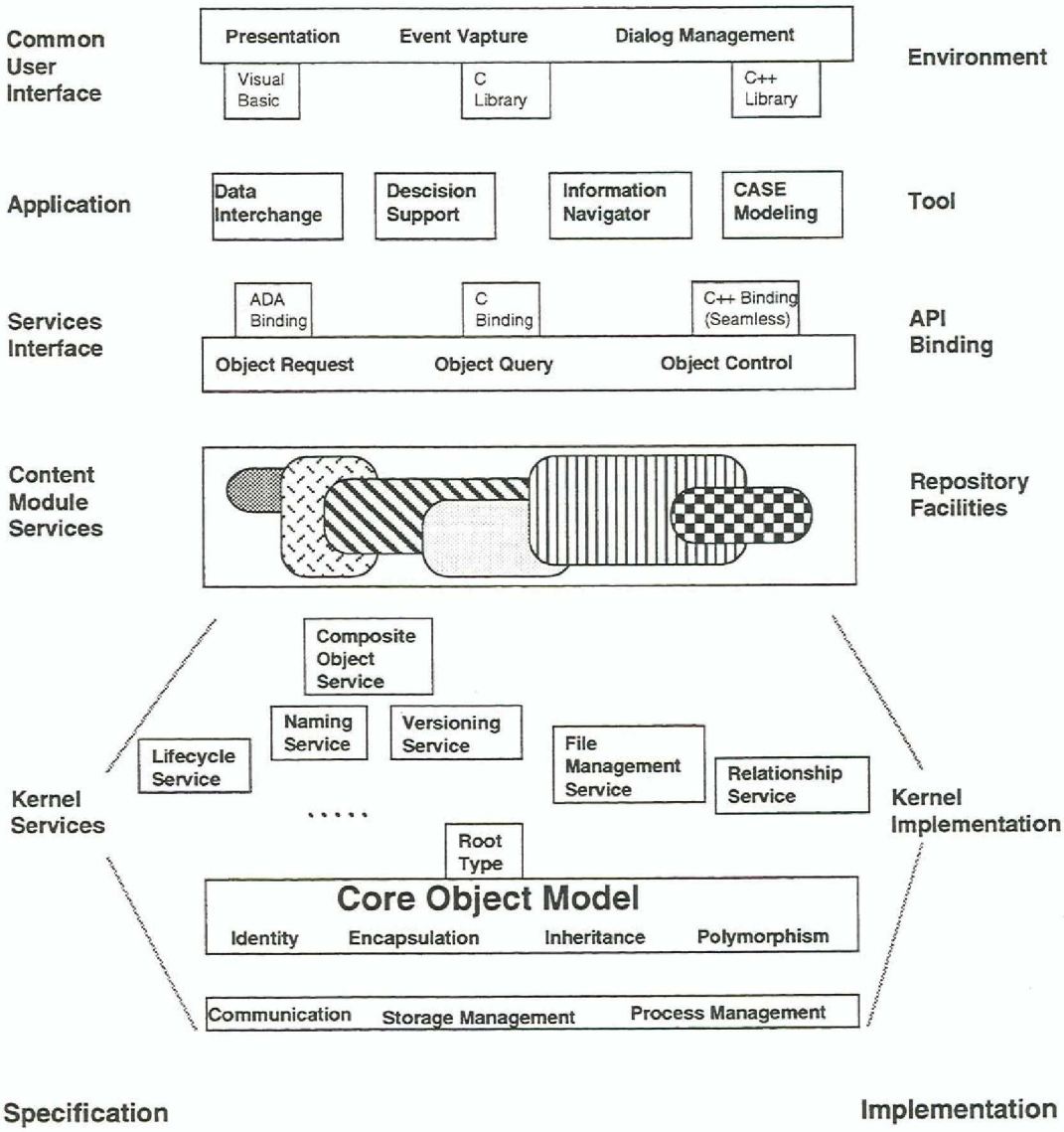
Figur 3

Nå, vad är då nytt i ANSIs nya ansats?

Jo, det nya är att man utgår från en metametamodell som tillåter specifikation av objektorienterade modeller (istället för ISOs SQL-modell eller ANSIs tidigare ER-baserade modell). Detta utnyttjar man på så vis att uppdelningen i operationer och det man opererar på inte längre ges samma relevans. Operationer eller services definieras som en del av objekttypsspecifikationerna. De flesta objekttyper är verksamhetsspecifika och ges därmed möjlighet att beskriva verksamhetsspecifika beteenden.

Dock kan ett antal verksamhetsobundna beteenden utkristalliseras, beteenden som måste (bör) vara tillgängliga för samtliga objekt, oavsett typ. Dessa generella "beteenden" (*Object Services*) inordnas som methods under generella objekttyper (root types). De verksamhetsorienterade objekttyperna formuleras sedan som specialiseringar av de generella.

Den generella delen, som egentligen också kan ses som en egen content module, formuleras i något som kallas en Core Object Model. Denna modells objekttyper har som dynamisk beskrivning tillsammantaget i huvudsak de beteenden som i tidigare ansatser utgjorde IRDS Services Interface. Figur 4 visar en ytterligare något kompletterad idéskiss.



Figur 4

Nästa avsnitt beskriver närmare den nya ansatsens syn på uppdelningen mellan Object Services och Services Interface.

3 IRDS Services

3.1 Services Architecture

Ett repository är i snäv mening en lagringsplats för diverse data inom ett visst tillämpningsområde. Tillämpningsområdet är i detta fall stöd i samband med arbete inom olika faser av ett systems livscykel. Ofta inkluderas även stöd vid utveckling av verksamheten som sådan. Stödet behövs av utvecklare, konstruktörer, kontrollanter, beslutsfattare, samordnare, ... Stödet erhålls i form av olika typer av funktioner eller services opererande på kunskap om system, verksamhet m m under olika faser av dess existens (historia). Denna kunskap representeras i form av tillståndsmodeller uttryckta i en överenskommen begreppsapparat. Enklare uttryckt; vi behöver kunna hantera (service) uppgifter (data) om något av intresse i en verksamhet och dess informationssystem (tillämpningsområdet). I grunden gäller samma problematik för de flesta databastillämpningar. Vad som gör IRDS något speciellt är behovet av ett antal typer av hanteringar som vanligtvis inte brukar återfinnas i ordinära databashanterare. Mer om det nedan.

Servicen byggs upp av byggstenar som tillsammans bildar en helhet. En arkitektur beskriver ingående byggstenar, deras beroendeförhållanden och sätt att samverka. Vilka byggstenar som behövs och hur de bör samverka kan man ha olika åsikter om. ANSI/X3H4 har levererat sin syn på saken i form av dokumentet 93-048R1.

I en repository services architecture ligger det nära till hands att anlägga ett dynamiskt perspektiv. Byggstenar kommer och går, samverkansprinciper förändras inom de givna ramar som arkitekturen specificerar. X3H4 använder sig i sin nya specifikation av en objektorienterad ansats för att beskriva byggstenarnas relevanta egenskaper och deras sätt att "umgås".

"Service" definieras som "A related set of operations that are invoked through the interfaces for one or more object types in response to requests or detected conditions".

3.2 Object Services

Objekten i ett IRDS behöver kunna hanteras på olika sätt. Dels finns där generella handgrepp, dels IRDS-specifika. Till de generella hör att skapa och radera objekt, förmodligen även att formulera utsökningar av delmängder av en objektmassa. Att kunna hantera flera versioner av ett objekt är däremot antagligen inte ett behov för alla typer av objekt, men väl för de flesta objekt i ett IRDS. Vanliga DBMS tillhandahåller i allmänhet de generella handgreppen. Dessa är numer ganska rikhaltiga och i ökande. Exempelvis finns funktioner för behörighet och accessrättigheter, för backup och recovery.

ANSI/IRDS nämner bland annat följande typer av service som erforderliga (Base Services). Listan kan göras mycket längre.

- Lifecycle service
- Name service
- Version service
- Event service
- Relationship service
- Security service
- Audit-trail service
- Composite object service

Base Services kan ses som ett bibliotek av allmänt tillgängliga beteenden inordnade under generella objekttyper inom Core Object Model.

Det finns inget som säger att all denna service ska utvecklas helt inom ett IRDS. Det kan inom Support Systems exempelvis finnas databashanteringssystem som klarar ett flertal av behoven. Existerande och tillgänglig funktionalitet används då lämpligen som byggstenar som helt enkelt anpassas och propageras upp genom arkitekturen.

3.3 Content Module Services

En content module består av en gruppering av de objekttyper (inklusive deras samband m m) som är av relevans inom visst hanteringsområde, som hantering av uppgifter inom viss utvecklingsfas av ett system, för att uttrycka en viss aspekt av ett system m m. Exempel kan vara processmodellering, informationsflödesmodellering, specifikation av distribuerade miljöer, EDI-specifikationer.

Utöver den generella servicen enligt föregående avsnitt kan det finnas behov av service som är riktad mot en specifik content module eller kanske mot någon verksamhetsspecifik objekttyp. Dit skulle exempelvis kunna höra olika typer av konsistenskontroller, händelsebaserade operationer och härledningar. De verksamhetsspecifika objekttyperna kan ingå ordinära sub/superklassförhållanden med andra objekttyper, bland annat lämpliga objekttyper från core object model. Genom arvsmekanismer omfattar de därmed vissa Base Services, något som underlättar formulering av önskad service över Services Interface.

3.4 Services Interface

Det totala schemat av intressanta objekttyper och deras samband finns beskrivna i den samlade mängden av content modules. Eftersom varje objekttyp innehåller såväl önskvärda statiska uppgifter som önskade hanteringsbeteenden (till viss del via arv) finns allt serverat och klart över Services Interface. Gränssnittsspråket (ett så kallat Application Programming Interface – API) är dock ännu inte definierat.

Låt oss skissa på ett exempel (med risk för feltolkning av intentionerna). Antag att repositoryt bland annat innehåller beskrivningar av olika relationsdatabasers scheman. En hantering eller operation skulle kunna vara att infoga det giltiga formatet för en viss kolumn i en relation. "Format" är en, av förmodligen ett antal, statiska egenskaper under objekttypen "Kolumn". Samtidigt har "Kolumn" genom arv från "Root Object Type" beteendet "Infoga" eller "Uppdatera egenskap". Över Services Interface begärs nu på lämpligt sätt att det aktuella objektets (kolumnen ifråga) beteende "Infoga" utförs med det aktuella formatet som parameter.

Vi kan krångla till det hela något genom att för objekttypen "Kolumn" kräva att varje uppdatering av dess egenskap "Format" ska föregås av en kontroll av formatets korrekthet. Över Services Interface begärs nu utförande av kolumnens lokalt definierade beteende "Infoga Format". Det utför först kontrollen efter sina lokala villkor. Därefter, under förutsättning att kontrollen utfallit till belåtenhet, aktiveras det från Root Object Type ärvda beteendet "Infoga".

Hur och var man ska formulera exempelvis en kontroll av att samtliga definierade relationer har en primär nyckel definierad, är mera oklart. Samma sak gäller generella utsökningar motsvarande vanliga frågespråk. Där är det ju vanligt att sökningen berör ett antal objekttyper. Här har vi just exempel på att det inte är alldeles naturligt att underordna data (statiska uppgifter) och operationer på data (beteenden) under samma "hatt". Trots allt är ju bearbetningsmodellen för ett IRDS och de verklighetsavbildande modellerna som beskrivs i ett repository olika saker – olika modeller. Varför blanda ihop dem? (Inte ens om det beskrivna i repositoryt är definitionen av ett IRDS, är det beskrivna och exekveringen av det beskrivna olika delar av samma modell. Eller?)

Ytterligare några frågetecken diskuteras i nästa avsnitt.

4 Några synpunkter

Den objektorienterade anpassningen har just påbörjats. Standarder kan inte påräknas inom den närmaste tiden. Technical Report revideras fortlöpande. I realiteten utförs arbetet av ett fåtal entusiaster med, som de bedömer, ett tyst bistånd av övriga. Tystnaden kan ju bero på att detta är en synnerligen komplex problematik som man inte orkar sätta sig in i ordentligt och i alla händelser inte känner sig beredd att debattera eller bemöta. En Technical Report är dessutom ett arbetsdokument och som sådant inte förbindande för framtiden.

Bland oklarheter kan nämnas:

- Operationer tycks normalt vara objekttypslokala. Samtidigt nämner man som exempel på services rapportfunktioner, generella frågespråksbaserade operationer, m m. Operationer inom viss context module kan ju mycket väl tänkas beröra en uppsättning objekt av samma eller olika typer, exempelvis en avancerad härledning eller konsistenskontroll.

- Frågan är om härledningar och konsistenskontroller m m ska ses som beteenden eller snarare, som är fallet i mer avancerade dbms, anges deklarativt i något formellt språk. Interpretering initieras osynligt i enlighet med förutbestämda regler, exempelvis i och med att den härledbara uppgiften begärs. Vad finns i så fall kvar som "riktiga" beteenden?

- Content modules kommer sannolikt att överlappa varandra. Samma objekttyp kommer alltså att hanteras med olika infallsvinklar. Vissa av dess beteenden kanske är av relevans inom en content module men inte en annan. Hur styra detta?

- Frågan är om service-indelningen lämpligen bör sträcka sig över content modules och objekttyper. Dessa kommer ju i ett realistiskt läge att vara underkastade ständiga modifieringar som inte alltid har inverkan på operationsuppsättningen. På motsvarande sätt kan det finnas behov av nya typer av operationer utan att innehållet i context modules påverkas. Borde inte sådana operationer ligga på tillämpningssidan snarare än i Services Interface. Om den frågan ibland kan besvaras med ja, hur ska man välja på "vilken sida" en ny funktion ska placeras?

- Vad är den egentliga skillnaden mot ett OODBMS kompletterat med ett specifikt schema, d v s att se IRDS-funktionalitet som en vanlig tillämpning? Visserligen saknas kanske vissa base services såsom audit-trail, men är skillnaden så stor att det motiverar dessa omfattande standardiseringsansträngningar?

- PCTE ligger ju väldigt nära i ansats genom alla specificerade operationstyper. Dessa motsvarar ju grovt IRDS services. Ett utökat samarbete mellan dessa båda riktningar vore önskvärd.

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
V 5: ...fråga är guld. Lokal affärsstyrning utifrån den egna verksamhetens data

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
N 10: Modelleringshandboken
1 – Översikt
2 – Modelleringsledarens bashandledning
3 – Modellering i grupp
4 – Kommunikation
5 – Arbetsgångar
6 – Modelleringsväskan
7 – Objektorienterad verksamhetsanalys
8 – Basmodeller
9 – Regelmodellering i praktiken
10 – Business Process Reengineering
11 – Namnsättning
12 – Tolkning av grafiska modeller
N 11: Ett+Ett=Ett – Två praktikers erfarenheter av modellintegrering

Kunskapsförmedling

H 1: Handledarutbildning för modelleringsledare, avancerad
H 2: Slutrapport HUMLA prototyp
H 3: Utbildning i Informationsadministration
H 4: Spridning av Hybris – en fallstudie vid Telia

Uttagssystem

U 1: Hybris i Unix-miljö
U 2: DEBRIS
U 3: Hybris DOS/PimWin på Posten
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: Koppling 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 I Information Model – Processer och och informationsflöden mellan processer
K 13: AD/Cycle I Information Model – Info Flows inom Processmodellen
K 14: AD/Cycle I Information Model – Relationsdatabasmodellering
K 15: AD/Cycle I 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
K 23: XLII – en öppen och flexibel utvecklingsmiljö
K 24: Hybris IA/DA – En IA-prototyp vid Telia
K 25: Introduktion till GDMO-standarderna
K 26: OpenODB
K 27: ANSI/X3H7 "Object Information Management"
K 28: Object Management Group
K 29: Översättning av modelldata
K 30: Objektorienterade ansatser inom ANSI/IRDS

KORT OM TRIAD

Triad är namnet på ett treårigt samarbetsprojekt kring informations-administration och dataadministration, IA/DA, som Telia, 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.