TRIAD Rapport K 22

PCTE
– en översikt

Stig Berild

SISU & SVERIGES TEKNISKA ATTACHÉER

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

LÄSANVISNING

Den läsare som endast önskar en allmän uppfattning om vad PCTE står för behöver bara läsa avsnitt 1, 2 och 5. De övriga avsnitten går mer in på detaljer.

1. Verktygsintegrering

För en utmärkt introduktion i ämnet hänvisas till Dan Nyström, Samverkan mellan resurskataloger, Triad-rapport K 11.

1.1 Olika typer av verktyg

De flesta större systemutvecklingsprojekt bedrivs idag med stöd av så kallade CASE-verktyg. CASE står normalt för Computer Aided Systems Engineering. Ibland får A-et betydelsen "Assisted" och E-et betydelsen Environment. Oavsett vilket, är den underliggande uttydelsen densamma, nämligen att med hjälp av dator och anpassade datorprogram ge systemutvecklare stöd och ibland även styrning i arbetet.

CASE-verktyg har av hävd ofta förknippats med:

- inriktning mot tidiga faser (analys, design)
- grafiska gränssnitt
- abstrakta modeller (av någon verklighet eller Universe of Discourse)
- orientering mot användare, d v s gränssnitt på slutanvändarens "språk"
- att specifikationen har ett egenvärde i sig och inte nödvändigtvis har programvara som mål

Arbetsuppgifter som stöds är bl a datamodellering, målmodellering, flödesmodellering, formulärmodellering och funktionsmodellering.

De verktyg som stödjer realisering ,och underhåll, av programsystem brukar också av hävd kallas programutvecklingsverktyg eller SE-verktyg, där SE är en förkortning av Software Engineering. Betoningen har lika mycket varit på underhåll och drift som på skapandeprocessen i sig.

Typiska egenskaper hos SE-verktyg är:

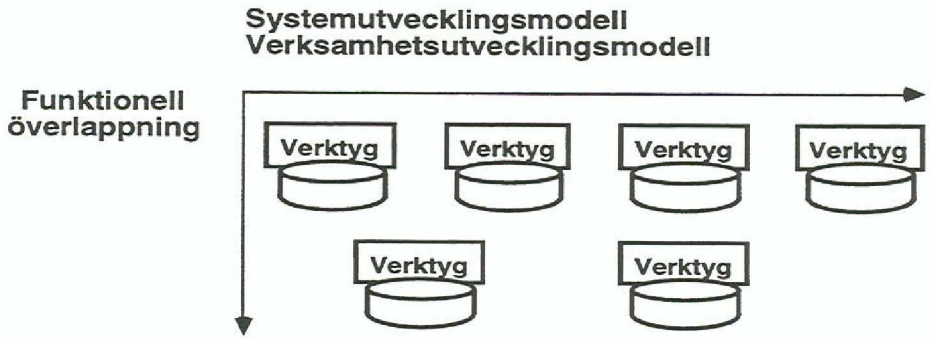
- inriktning mot senare faser (realisering, test, underhåll)
- programvaruframställning
- metod -och struktureringsstöd
- programkörning
- stöd för hantering av olika versioner och kombinationer av beskrivningselement
- expertorienterade, d v s verktyget är främst avsett för utvecklare

Arbetsuppgifter som stöds är bl a programstrukturering, redigering, avlusning, kompilering, konfigurering.

Caseverktyg blir alltmer heltäckande, SE-verktyg likaså. Processen att utveckla, driftsätta och underhålla system blir allt mindre entydigt fas-indelat (vattenfallsprincipen). Istället flyter faser i varandra, återupprepas o s v. I och med att perspektivet alltmer läggs mot ett systems totala levnad, suddas indelningen i kategorier ut ytterligare. För det fortsatta resonemanget saknar uppdelningen i CASE- och SE-verktyg relevans. Dock är det viktigt att klargöra skillnaden för att lättare förstå målsättning för och egenskaper hos PCTE.

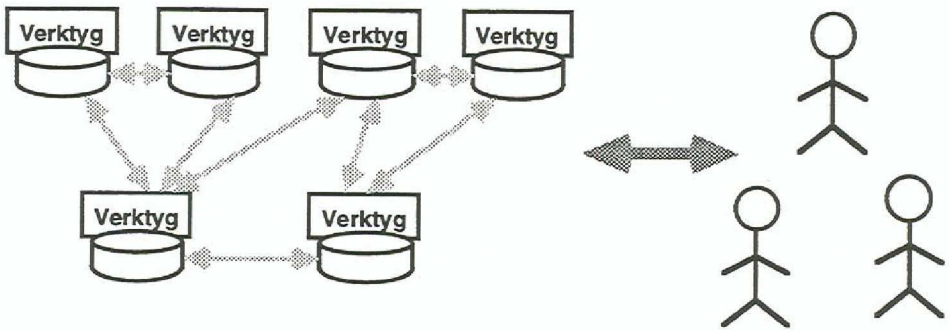
1.2 Behov

Vi kan konstatera att hantering av hela eller delar av ett systems livscykel kräver många olika typer av stöd och funktioner, packeterade i ett eller flera verktyg. Ju mer heltäckande behov, desto osannolikare att finna ett enda verktyg, som svarar mot dem alla. Ju större organisation arbetet bedrivs inom, desto svårare att upprätthålla samordning och strategier. Olika verktyg används för samma, överlappande eller olika typ av stöd. Se figur 1.



Figur 1

Används flera verktyg inom samma projekt, är det sannolikt att de måste kunna samverka. Samverkan behövs säkert också på en övergripande nivå för att säkerställa återanvändning, gemensamma policies, funktionella och dataorienterade beroenden osv. Inte minst måste de kunna ge stöd till delvis samma verktygsanvändare. Se figur 2.



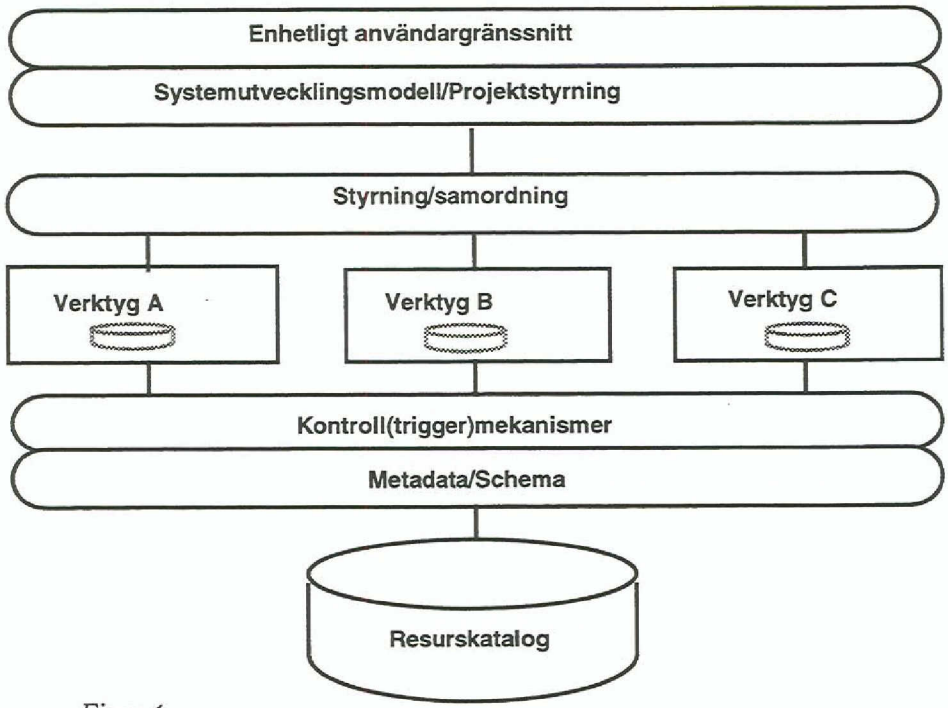
Figur 2

1.3 Olika grad av integrering

Samverkan ställer krav, ju högre krav desto mer av anpassning, samordning, styrning. Av den anledningen har olika ansatser vuxit fram för olika behov och förutsättningar. En enkel ansats är att skicka data mellan verktyg i enlighet med en överenskommen syntax och semantik, figur 3. Samordningen innebär endast att mottagaren kan förstå vad avsändaren har skickat. Samordningen gäller endast data och endast tillfälligt i samband med en överföring. CDIF-standarden (beskriven i Triad-rapport K 22) är av denna typ.

Figur 3

En betydligt högre grad av integrering visas i figur 4.



Figur 4

Några kommentarer:

- Samtliga verktyg arbetar med data i en och samma logiska databas. (Om den rent fysiskt är distribuerad eller ej är inte relevant.) Denna databas kallas fortsättnings vis för resurskatalog. I engelskspråkig litteratur används begreppet "repository" (ofta även i svenska texter, i brist på vedertaget begrepp). (Resurskatalog)
- Innehållet styrs av ett gemensamt schema. Samtliga verktyg förutsätts ställa upp på schemats utformning. (Metadata/schema)

- I det generella fallet kan det behövas en bevakning av vilka typer av operationer och/eller vilka data som hanteras hur och när i resurskatalogen. Vissa händelser kan på något sätt vara av intresse för eller beröra ett verktyg. Verktyget behöver av den anledningen bli informerad om det inträffade. (Kontrollmekanismer)

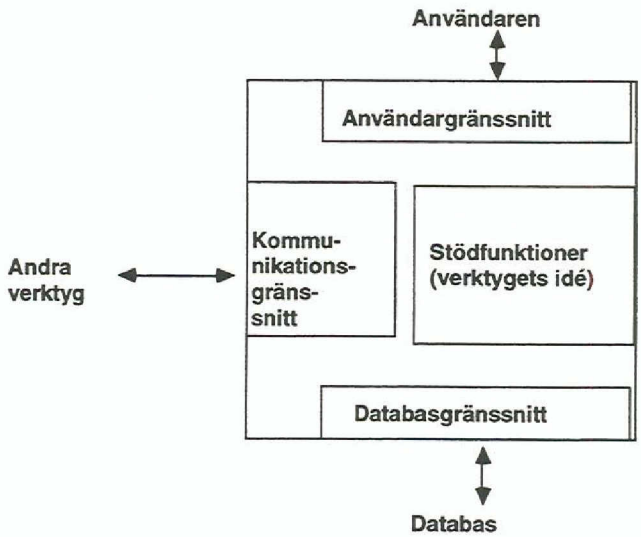
- Vissa verktyg kan arbeta mer eller mindre intimt tillsammans. De kan utnyttja funktioner hos varandra istället för att utveckla egna motsvarande. De kanske

tillsammans utför en sammanhängande funktion osv. Mekanismer för detta samarbete måste finnas. (Styrning/samordning)

* På ett högre plan kan förekomma krav på utveckling i enlighet med en given systemutvecklingsmodell. Detta ställer krav på samordning mellan verktygen, exempelvis vilka som får/ska göra vad, när och hur, inklusive uppföljning och kontroll. (Systemutvecklingsmodell/Projektstyrning)

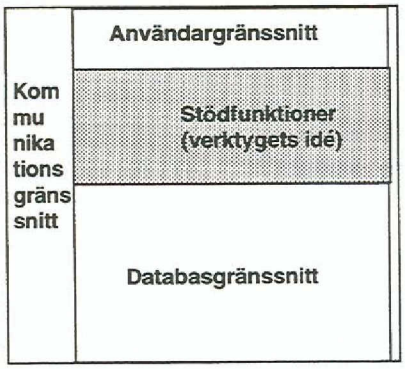
* Vi har tidigare konstaterat att de olika verktygen antagligen kommer att användas av delvis samma människor. Deras arbete underlättas om de kan tillämpa samma principer och symboler oavsett vilket verktyg de för tillfället arbetar med. (Enhetligt användargränssnitt)

Sett ur verktygets perspektiv måste det, förutom de egna unika egenskaperna, även finnas funktioner som utför och stöder åtminstone tre gränssnitt mot omvärlden; mot resurskatalog, mot andra verktyg och mot användare. Se figur 5.

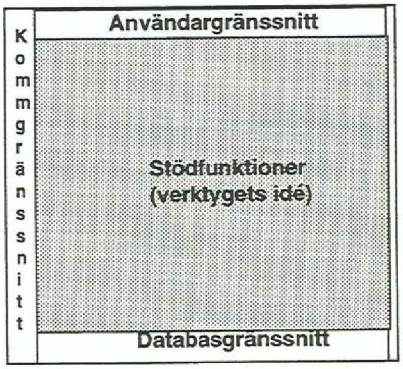


Figur 5

Ökad integrering innebär normalt också att funktioner renodlas. I dagsläget omfattar dock de flesta verktyg var och en av dessa funktioner, förutom dess huvuduppgift. Inga eller ytterst få verktyg har medvetet utvecklas som en "ruta" i figur 4, förmodligen för att integreringsområdet som sådant i många delar saknar både stabilitet och standarder. Följden blir att varje verktygsutvecklare måste lägga ner ansenliga resurser på faciliteter som i idealfallet skulle finnas som separata, färdiga och anropsbara stödfunktioner. Figur 6 a visar dagsläget, figur 6 b den önskade situationen. Observera, att de flesta verktygsleverantörer är ganska små företag. Utrymmet för nytänkande blir begränsat så länge de måste "göra allt".



Figur 6 a



Figur 6 b

PCTE har utvecklats med syftet att erbjuda en heltäckande standardlösning i form av specifikation av ett standardiserat data- och processgränssnitt. Givet en allmänt accepterad standard av denna typ blir varje verktyg enkelt möjligt att ansluta till olika konfigurationer och till andra verktyg, utan behov av anpassningar eller begränsning till vissa hårdvaror och operativsystem. Tyvärr är detta än så länge mer vision än verklighet.

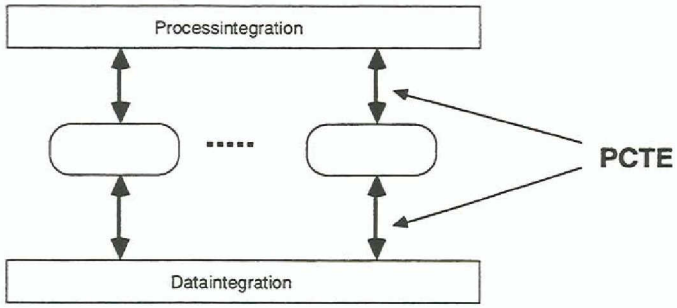
2. Översikt av PCTE

2.1 Målsättning

PCTE är en förkortning av Portable Common Tool Environment. Det startade som ett Esprit-projekt 1983 och pågår, tio år senare, fortfarande med god intensitet. Figur 8 visar några historiska faser.

Målsättningen har varit att definiera och testa ett gränssnitt för ett generellt Software Engineering Environment (SEE). För att återkoppla till inledningen avses mao stöd framförallt för programutvecklingsfaserna. Stödet ska vara oberoende av språk, klara en distribuerad utvecklingsmiljö och ha en primär orientering mot LAN och arbetsstationer. Vetskapen om att det redan existerar ett antal verktyg, och att dessa utvecklats utan tanke på PCTE, har fört fram tilläggskravet att befintliga verktyg utan större möda ska kunna inordnas under PCTE-gränssnittet. Gränssnittets repertoar ska vara så heltäckande att i princip alla erforderliga data- och processoperationer ska kunna formuleras. PCTE blir näst intill ett operativsystemgränssnitt.

Det bör återigen poängteras att fokus ligger på process- och dataintegration. Se figur 7. Användargränssnitt bedöms i och för sig vara av vikt att standardisera. Samtidigt konstaterasi olika dokument att det finns andra grupper och etablerade de facto-standarder som är bättre skickade för denna aspekt.



Figur 7

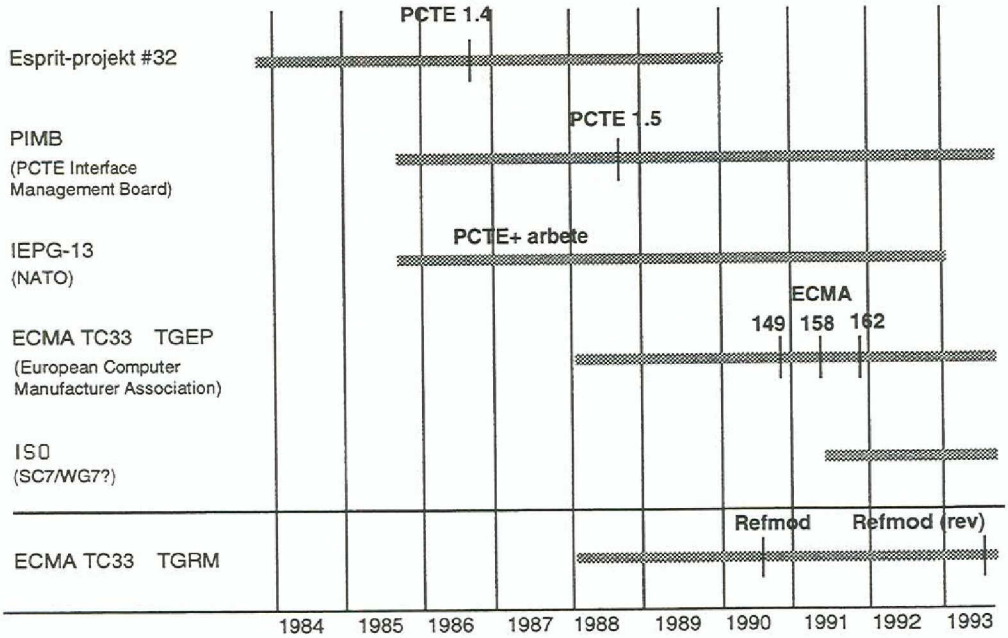
Vikten av att det finns överenskomna scheman noteras alltmer liksom behovet av att i resurskatalogen kunna hantera mer "fine-grained objects" (se vidare kapitel 6, nedan).

2.2 Historia, nuläge

Några kommentarer till händelseförloppet genom åren:

- Den första formella PCTE-definitionen kom i september 1986 under beteckningen PCTE 1.4. UNIX-inflytandet är här uppenbart.

- PIMB (PCTE Interface Management Board) startades av bl a NATO och ett antal större företag för att utvärdera och vidareutveckla PCTE 1.4. I november 1988 släpptes PCTE 1.5. Den innehåller mindre justeringar och rättelser av PCTE 1.4.

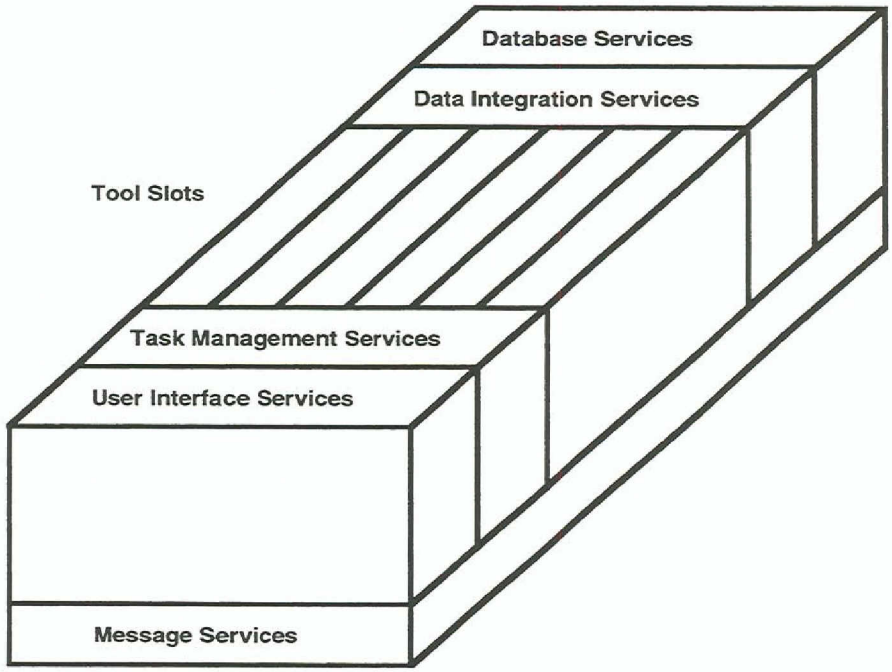


Figur 8

- Ungefär samtidigt startade NATO IEPG-13 (Independent European Program Group Technical Area 13) med syfte att komplettera PCTE med specifika försvarsorienterade egenskaper. Till dessa hör:
  – oberoende av operativsystem
  – bättre säkerhets- och behörighetsmekanismer
  – enhetlig hantering av schema och data, d v s i samma databas
  Arbetsresultaten går under beteckningen PCTE+. Arbetet har nu avslutats. PCTE+ har haft stort inflytande på innehållet i ECMA PCTE (se nedan).

- ECMA (European Computer Manufacturer Association) består av ett antal företag med tillverkning i Europa. Förutom typiskt europeiska företag (exv Ericsson, Televerket, Bull, NOKIA) ingår även ett antal utomeuropeiska företag med tillverkning i Europa (t ex HP, IBM, Dec, Sony). Vid årsskiftet 1987/88 tillsatte ECMA Technical Committee 33, TC33, för att fortsätta PCTE-arbetet. TC33 etablerade två arbetsgrupper:
  – TGRM (Technical Group on the Reference Model)
  – TGEP (Technical Group on ECMA PCTE)

- TGRM syftar till att ta fram en generell referensmodell för hur olika delar av en integrerad miljö hänger ihop och vad de bör klara av. Den första versionen av en referensmodell presenterades 1990. Intresset kring och behovet av referens-modeller har vuxit alltmer. Många synpunkter och försök att applicera referensmodellen på existerande produkter har resulterat i revideringar av det ursprungliga dokumentet. Tillsammans med amerikanska NIST ISEE Working Group (NIST= National Institute of Standards and Technology, ISEE=Integrated Software Engineering Environment) har TGRM just tagit fram en tredje version av ett referensmodelldokument. Dokumentet har vuxit från ca 50 sidor till nu mer än det dubbla. Referensmodellen är känd och ofta refererad. De flesta känner väl till "the toaster model", en idéskiss kring integrering (som ofta missförstås som en arkitekturskiss). Se figur 9.



Figur 9

- Syftet med TGEP var och är att fortsätta vidareutvecklingen av PCTE-standarden. En abstrakt gränssnittsdefinition som är oberoende av operativsystem presenterades i slutet av 1990 (ECMA-149: Abstract Specification). Strax därefter kom dels en specifikation av hur den abstrakta specifikationen ska appliceras i språket C (ECMA-158: C Programming Language Binding), dels en motsvarande specifikation för Ada (ECMA-162: Ada Programming Language Binding). Samtliga tre dokument är för närvarande föremål för remissarbete och revidering. Nya versionerna förväntas bli tillgängliga sommaren 1993.

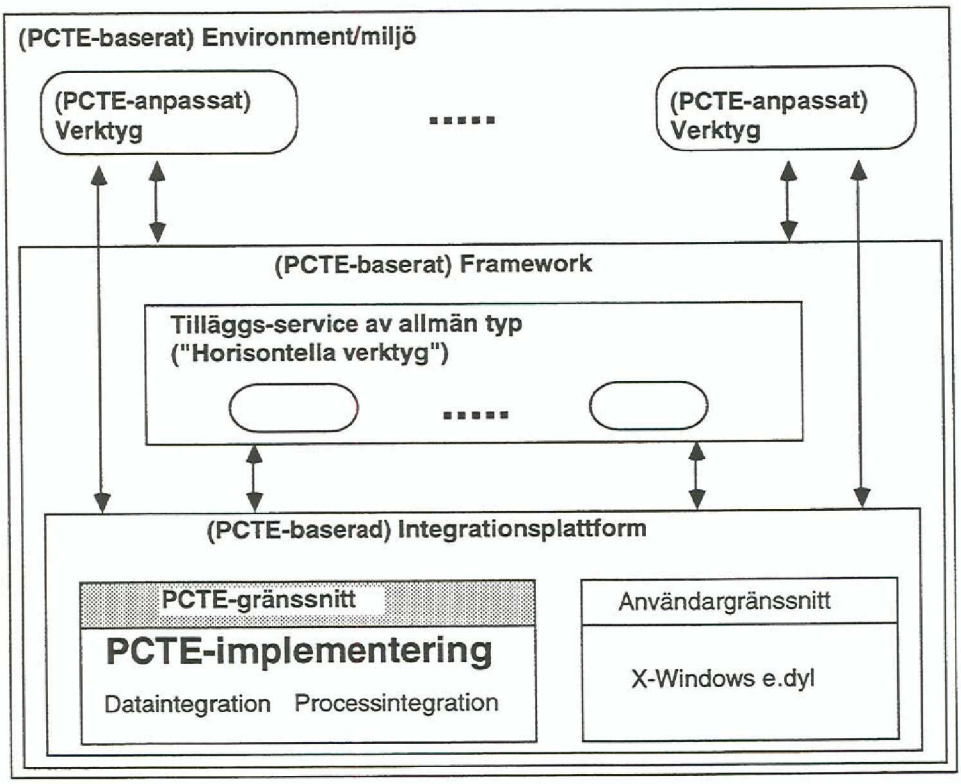
- Förutom ovanstående revideringar pågår arbete med bindning till C++. En förhoppning är att de nya dokumenten ska accepteras av ISO för snabbbehandling av lämplig Sub Committee till internationell standard. ISO har än så länge antytt ett välvilligt intresse.

2.3 PCTE:s plats i en integrerad miljö

Som redan nämnts är PCTE en gränssnittsdefinition. Detta gränssnitt formulerar alla nödvändiga basoperationer (ett minsta antal för full funktionalitet). En realisering i syfte att kunna tolka gränssnittet och utföra motsvarande operationer, inklusive lämpligt användargränssnitt, brukar gå under benämningen *PCTE Plattform.*

Nästa servicenivå är *PCTE Ramverk* (Framework). Förutom plattformen ingår ett antal generella påbyggnadsfunktioner med syfte att svara mot förmodat vanliga och överensstämmande behov hos ett antal Case-verktyg. Verktygen kan formulera vissa av sina behov mot dessa funktioner på högre nivå istället för att formulera dem som mer eller mindre komplexa kombinationer av operationer direkt mot PCTE-gränssnittet. Exempel på en påbyggnadsfunktion är en frågespråkshanterare som opererar med ett mer sammansatt, effektivt och lättanvänt språk, än de basoperationer som erbjuds direkt i PCTE-gränssnittet.

Ett PCTE Ramverk kompletterat med de CASE-verktyg som avses ge de aktuella användarkategorierna ett fullgott arbetsstöd går under namnet *PCTE Miljö* (Environment). CASE-verktygen brukar ibland kallas vertikala verktyg därför att de klarar ett behov, medan de generella påbyggnadsverktygen kallas horisontella verktyg eftersom de ger service åt många). Se vidare figur 10.



Figur 10

3. Resurskatalogen

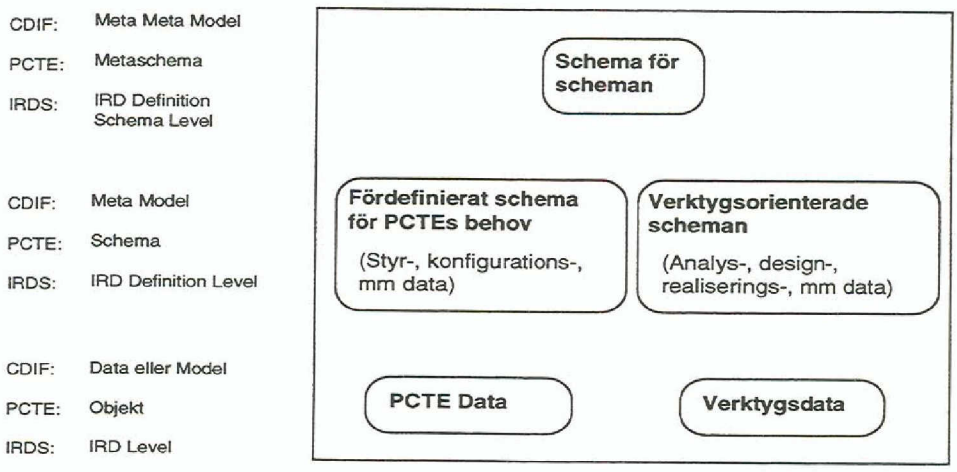
3.1 Roll och innehåll

En vital del av PCTE:s grundfilosofi är att hantera alla erforderliga specifikationer av ett system under alla intressanta delar av dess livscykel i en gemensam resurskatalog. Innehållet ska vara schemastyrt. En totalsamordning under ett enda schema anses orealistiskt med hänsyn till den stora innehållsbredd PCTE avses täcka. Viss samordning är både möjlig och eftersträvansvärd, men är för närvarande utanför PCTE:s ramar, med vissa undantag. Istället avser man etablera olika typer av samarbete kring resurskatalogens innehåll, exempelvis med CDIF.

Som undantag finns ett antal fördefinierade delscheman, dels i form av en övergripande grundstruktur, dels i form av scheman som PCTE själv behöver för att fullgöra sina egna arbetsuppgifter (exempelvis behörighet, låsning och konfigurering). Därmed är vi inne på resurskatalogens roll och innehåll. I en traditionell syn skulle de data som behövs för att driva PCTE-tillämpningen ses som åtskilda de data som PCTE-tillämpningen hanterar åt verktygsanvändarna. Inom PCTE läggs de dock i samma databas (resurskatalog). I en traditionell ansats ligger schema åtskilt från de data schemat strukturerar. Inom PCTE ligger data som beskriver schemat i samma resurskatalog som de data schemat strukturerar. Eftersom både de data verktygen hanterar och PCTE-driftsdata ligger i samma resurskatalog, kommer respektive schema att också återfinnas i samma resurskatalog.

Schemadata är också data som behöver arrangeras enligt ett schema. I konsekvensens namn måste även detta metaschema läggas in i resurskatalogen. Så skulle man kunna resonera i oändlighet. Dock kan man stanna vid metaschemat eftersom schemat för att beskriva metaschemat också är ett metaschema som stämmer överens med "sig själv".

Kontentan av resonemanget är att allt ligger i samma resurskatalog och kan hanteras på ett enhetligt sätt. Principen innebär både för- och nackdelar. Det ligger dock utanför denna rapport att analysera detta närmare. Figur 11 illustrerar resurskatalogens innehåll enligt ovan. I vänstermarginalen återfinns de begrepp, som olika standardiseringsansatser använder för respektive modellnivå. Någon samstämmighet i terminologin finns inte. IRDS finns beskriven i TRIAD-rapporterna K1 och K2 samt CDIF i K 21.



Figur 11

3.2 Principer för datahantering och dataintegration

Ett visst verktyg behöver tillgång till ett antal olika delscheman för olika behov och funktioner. Ett sådant delschema kallas för ett *Schema Definition Set (SDS)* och omfattar alltså en viss uppsättning schemakomponenter för att täcka in ett avgränsat syfte. Ett SDS kan vara ett delschema för att hantera flödesmodeller, ett annat kan finnas för att formulera programstrukturer, ett tredje för att i detalj beskriva dataflödens struktur och innehåll inom en flödesmodell o s v. Inför en körning av ett visst verktyg (kallas process) pekas de SDS ut som kommer att behöva nyttjas under den aktuella processen. Refererade SDS kallas med ett gemensamt namn för processens *Working Schema*. Nästa gång verktyget körs kanske en annan uppsättning SDS är aktuella o s v.

Finessen med dessa SDS är dels att de förhoppningsvis står för naturliga grupperingar av schemakomponenter, dels att de är generellt definierade och därmed tillgängliga för flera verktyg. Där kommer integreringsaspekten in. Anser två eller flera verktyg att ett visst SDS för ett visst syfte beskriver de data som ska hanteras på ett acceptabelt sätt, utnyttjar de alla samma SDS (under förutsättning att inga behörighetsrestriktioner finns). Att två verktyg är 100% överens om hur exv flödesmodeller ska modelleras är å andra sidan inte så troligt, i alla fall inte innan området stabiliserats. Ett sätt att lösa dilemmat är att låta varje verktyg etablera sin version. Den givna nackdelen blir att verktygen hanterar sina data, visserligen i samma resurskatalog men med "vattentäta skott" emellan. Idén med en resurskatalog försvinner. En bättre lösning vore att göra det gemensamt som man är överens om och hantera resten separat.

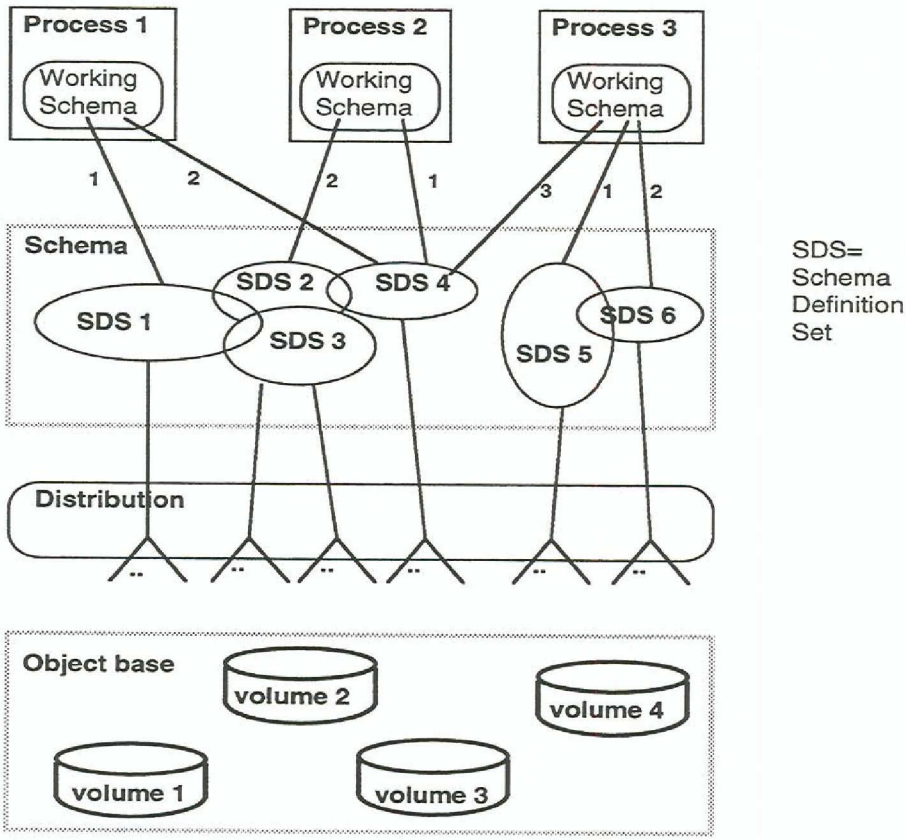
PCTE innehåller funktioner för denna lösning. Antag att SDS "A" finns definierat. Vid skapandet av SDS "B" konstateras viss överlappning med A. Integreringen åstadkoms på så vis att i B återanvänds de delar av A som överensstämmer. Det är alltså inte fråga om en förhandlingsprocess mellan de ansvariga för A och B som måste mynna ut i en kompromiss i form av det gemensamma schmat. B får ta det som bjuds och som man ser en fördel i att samordna. Inget hindrar dock givetvis B från att försöka ta kontakt med A för diskussion om en modifiering av A för att bättre passa B. På så vis har man skapat största möjliga samordning, utan tvång. Som tidigare nämnts eftersträvas inte någon total samordning till ett gemensamt schema. Hur hanteringen av ett SDS går till mer i detalj beskrivs i avsnitt 3.4.

En resurskatalogs totalschema blir "unionen" av samtliga definierade SDS.

Ett verktygs totalschema blir "unionen" av samtliga SDS det i något sammanhang önskar tillgång till.

Ett verktygs schema i samband med en viss körning (Working Schema) blir "unionen" av de SDS som processen kan komma att behöva nyttja.

Principen åskådliggörs i figur 12. Där framgår också att en resurskatalog kan vara distribuerad, något som är helt osynligt för de verktyg som opererar på den. I verktygets perspektiv är resurskatalogen en enda logisk enhet.



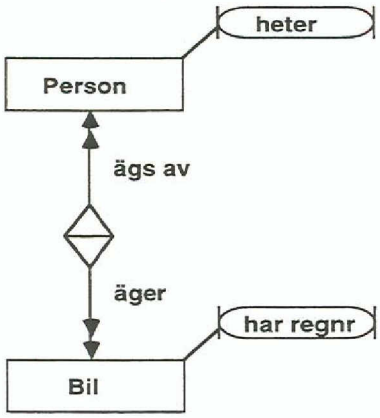
Figur 12

3.3 Modellnotation

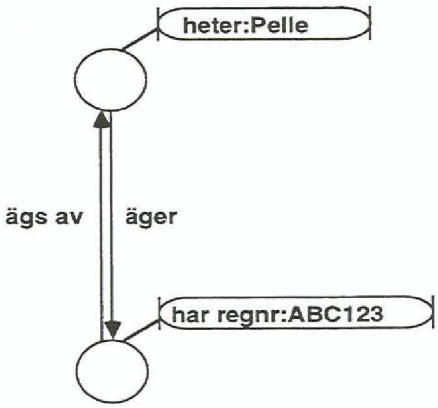
Innan vi går in på uppbyggnaden av metaschemat och den fördefinierade delen av schemat, introduceras här den grafiska notation, som ofta används i PCTE-sammanhang. Notationen är inte en del av standarden men har etablerats som ett "de facto-språk". Förutom att visa de olika grafiska symbolerna, är syftet med detta avsnitt att introducera den uttryckskraft som finns tillgänglig, d v s de begrepp och den semantik som finns representerad i metaschemat.

Antag det vanliga exemplet med personer, bilar och ägarförhållande (ett för PCTE:s tillämpningsområde något konstigt exempel, men försök bortse från det). Figur 13 a visar ett förenklat schema.

Objekttyperna (Object type) symboliseras med rektanglar. Attributtyperna (Attribute type) hänger vid sidan om i form av rektanglar med rundade hörn. Ägarsambandet symboliseras genom en dubbelriktad pil, i mitten kompletterad med två trianglar som pekar åt varsitt håll. Varje pilriktning (Link type) har sitt namn och sin semantik. Enkelpil indikerar ett en-förhållande, dubbelpil ett många-förhållande. Figur 13 b visar hur en motsvarande enkel förekomstnivå kan se ut. Notera att sambandet är uppdelat i två fristående delar (links). Syftet med att ha varje riktning som fristående företeelse kommer snart att framgå.



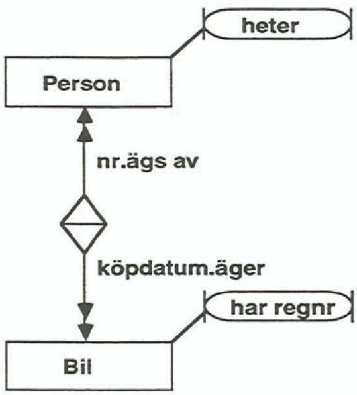
Figur 13 a



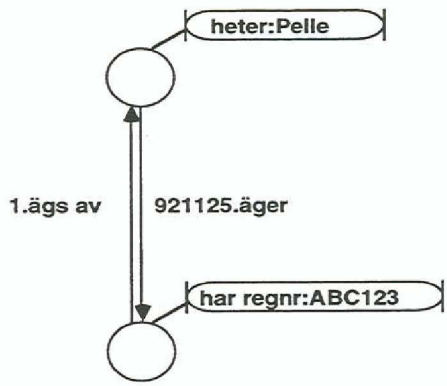
Figur 13 b

Modelleringsbegreppen har skapats i syfte att kunna formulera datamodeller på ett Entity-Relationship-liknande sätt. Minst lika vägledande har dock varit principen för referens till objekt i resurskatalogen. Referens till objekt sker inte genom att applicera en intern nyckel eller en unik kombination av attribut. Det sker alltid genom en mer eller mindre komplicerad navigering över existerande links. Dessutom finns inga mängdoperationer. En navigering resulterar alltid i referens till ett enda objekt (eller inget).

För att kunna söka ett visst objekt över en pilriktning (link) måste denna aktuella link entydigt identifieras. Det räcker inte att starta vid "Pelle" och navigera över "äger" eftersom det på andra sidan kan finnas många bilar (mängdoperation). Vi måste vara mer precisa, exempelvis genom att begära Pelles bil nr 1, Pelles bil med registreringsnummer ABC 123 eller den bil Pelle köpte 930525. I PCTE väljer man en lämplig sådan princip och uttrycker den som ett prefix till namnet på aktuell link type. I figur 14 a har vi exemplifierat med *köpdatum*-alternativet för *äger* och, i brist på bättre, med ett löpnummer *nr* för *ägs av*. Det bör påpekas att prefix bara behövs när det är fråga om många-förhållanden. Vi återkommer till navigering i avsnitt 3.5.



Figur 14 a

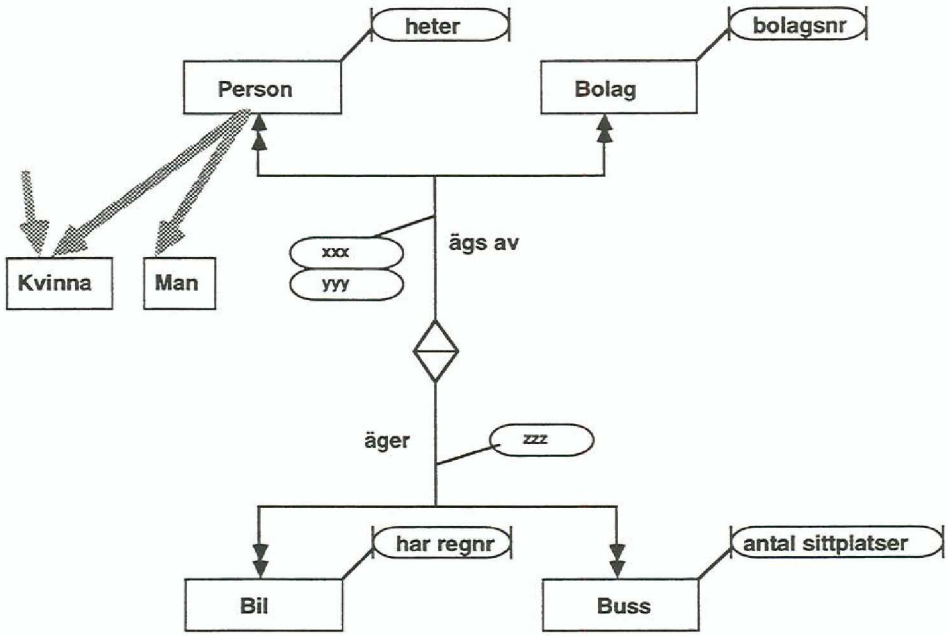


Figur 14 b

Varför nu denna ovanliga princip, kan man fråga? Modellen är ca tio år gammal. Unix-influensen finns där, likaså den historiska orienteringen mot stöd framförallt för programutveckling. I dessa sammanhang arbetar man ofta med ett objekt i taget. Objekten är också av typen coarse-grained, med fil eller liknande som minsta byggsten.

Sammanfattningsvis, metaschema och schema formuleras i enlighet med figur 14 a medan förekomstnivån har sin egen notation, som i figur 14 b.

Nu över till en utökad version av samma schema i figur 15.



Figur 15

I figur 15 är följande nytt:

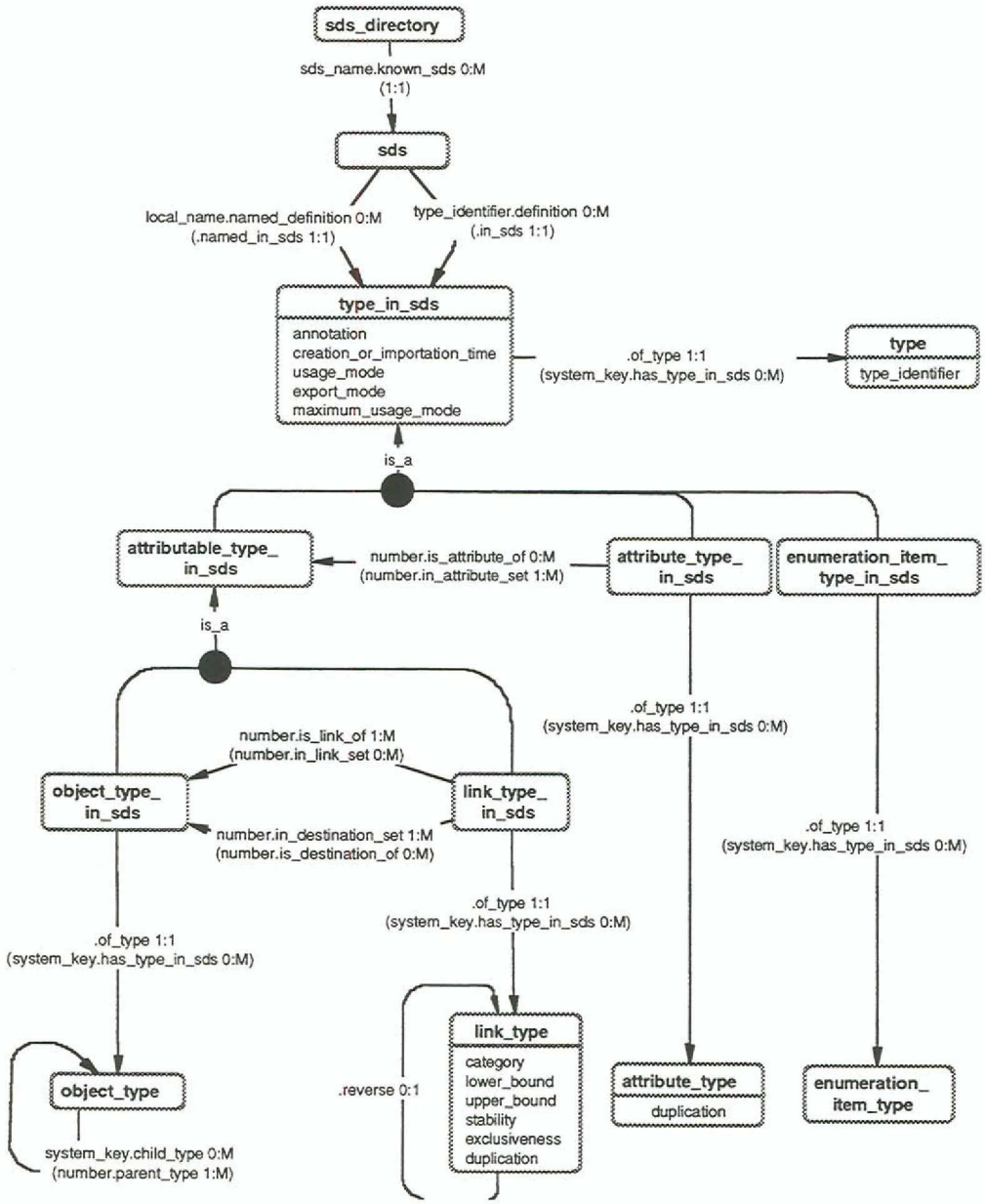
- De tjocka pilarna indikerar subtypsförhållande. Vid "Kvinna" kan vi se att multipla supertyper är tillåtna och därmed mekanismer för s k multipla arv.

- I vardera änden av en link type kan finnas mer än en object type. Både Person och Bolag kan äga såväl Bil som Buss. Inga extra (artificiella?) supertyper som Ägare och Fordon behöver skapas på varje sida. Objekttyper kan tillföras allteftersom förutsättningarna förändras. Därav följer indirekt att varje link type har en fristående existens oavsett om object types finns kopplade till den eller vilka object types som finns, vilket är något ovanligt. Samma sak gäller attribute types, som i och med detta snarast är att betrakta som domäner enligt traditionellt synsätt.

- Det kanske mest orginella är möjligheten att beskriva varje link type med en egen uppsättning attribute types. I traditionella sammanhang tillåter man ibland sambandstyper att ha egenskaper, men här har alltså varje riktning sin uppsättning! Det tidigare nämnda navigeringsmotiverade prefixet till link type-namnet är i själva verket att se som en sådan attribute type med en speciell roll.

3.4 Meta-schema

Metaschemat kan mycket väl uttryckas med metaschemats egen begreppsapparat och PCTE-notation. Av praktiska skäl och för att relatera PCTE till en inom Triad känd grafisk notation har jag i figur 16 valt BM-notationen. Den klarar de formuleringar som behövs för att uttrycka metaschemat. (Däremot är den inte lika lämpad för schema-nivån.) Namnen på sambandstyperna har kompletterats med aktuella prefix. I allmänhet används en sekvensnumrering (*number*) eller en systemgenererad nyckel (*system_key*). Observera att figuren är ett försök till tolkning av standarden efter bästa förstånd. Fel eller ofullständigheter kan förekomma.



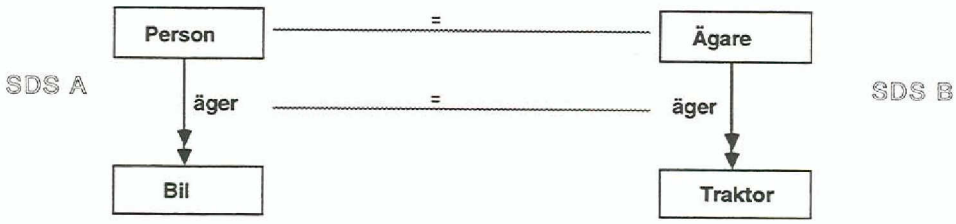
Figur 16

Ett lokalt SDS-schema byggs upp med hjälp av den övre delen av metaschemat, dvs allt som har suffixet "in_sds". Detta motsvarar grovt en vanlig metamodell, exempelvis CDIF:s eller Telmods, med de varianter som beskrivits ovan. På denna nivå sker ingen delning/integration mellan SDS. Integrationen ombesörjs av den nedre radens objekttyper enligt följande:

- För varje object_type_in_sds, link_type_in_sds, attribute_type_in_sds och enumeration_item_type_in_sds, som skapas inom ett SDS och för vilka ingen integrering är av intresse (eller har givits någon reflexion), skapas ett motsvarande generellt objekt av typen object_type, link_type, attribute_type respektive enumeration_item_type.

- Vill man däremot uttrycka att exempelvis en viss object_type_in_sds i SDS "A" är densamma som en annan object_type_in_sds i SDS "B" begär man en import av aktuell object_type_in_sds från "B". Internt redovisas detta genom att en ny object_type_in_sds skapas inom "sitt" SDS, d v s "A" samtidigt som det sätts att peka på samma object_type som dess motsvarighet i "B" pekar på.

Ta ett enkelt exempel.



Figur 17

SDS A har schemat till vänster i figur 17, SDS B det till höger. A finns när B skapas. Den som definierar B konstaterar att den object_type som kallas *Person* i A är densamma som i B kallas för Ägare. Respektive link_type *äger* överensstämmer inte bara till namn utan även i betydelse. Det finns behov av samordning. B-modellören gör följande:

SDS_CREATE_OBJECT_TYPE (SDS B, Traktor)

– Skapar en ny object_type till SDS B. Ingen tanke på integration.

SDS_IMPORT_OBJECT_TYPE (SDS B, SDS A, Person, Ägare)

– "Importerar" eller återanvänder den object_type som i SDS A kallas *Person* till SDS B samt ger den det lokala namnet *Ägare*.

SDS_IMPORT_LINK_TYPE (SDS B, SDS A, äger)

– "Importerar" eller återanvänder den link_type som i SDS A kallas *äger* till SDS B. Samma namn användes.

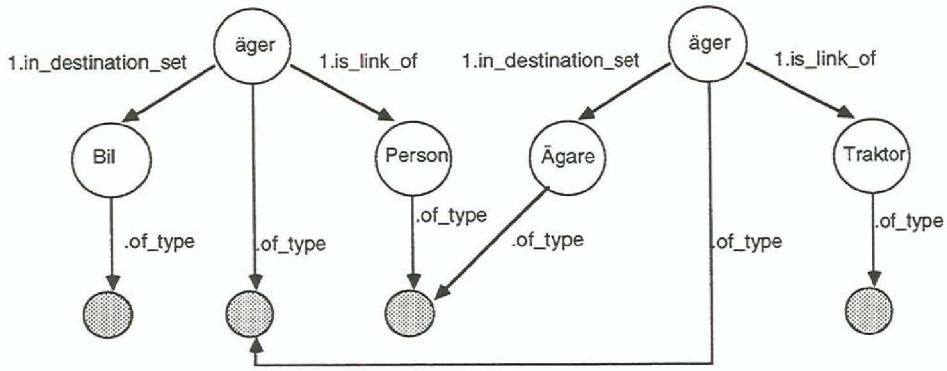
SDS_APPLY-LINK_TYPE (SDS B, äger, Ägare)

– I SDS B utgår nu link_type *äger* från object_type *Ägare.*

SDS_ADD_DESTINATION (SDS B, äger, Traktor)

– I SDS B går nu link_type *äger* till object_type *Traktor.*

Resultatet på förekomstnivå visas i figur 18.



Figur 18

Några fler kommentarer kring metaschemat:

- I den generella delen återfinns också en del egenskaper som bedömts vara av en från SDS fristående natur.

- Som synes ingår i den generella delen endast meta-typerna, inte hur de är relaterade genom eventuella samband. Sambandstyperna anses alltid vara beroende på hur och var meta-typerna används, d v s de tillhör ett visst SDS. Detta är möjligt eftersom link types kan ha en av object type oberoende existens. Alternativt har man tvingats göra dem självständiga för att inte integreringsmekanismen ska bli för komplex.

- Object_type kan ha en "content type". Därmed antyds att det finns en intern struktur av intresse. PCTE erbjuder ett visst, begränsat stöd för operationer på "contents".

- Attribute_type är alltid av viss value type och har ett initialvärde.

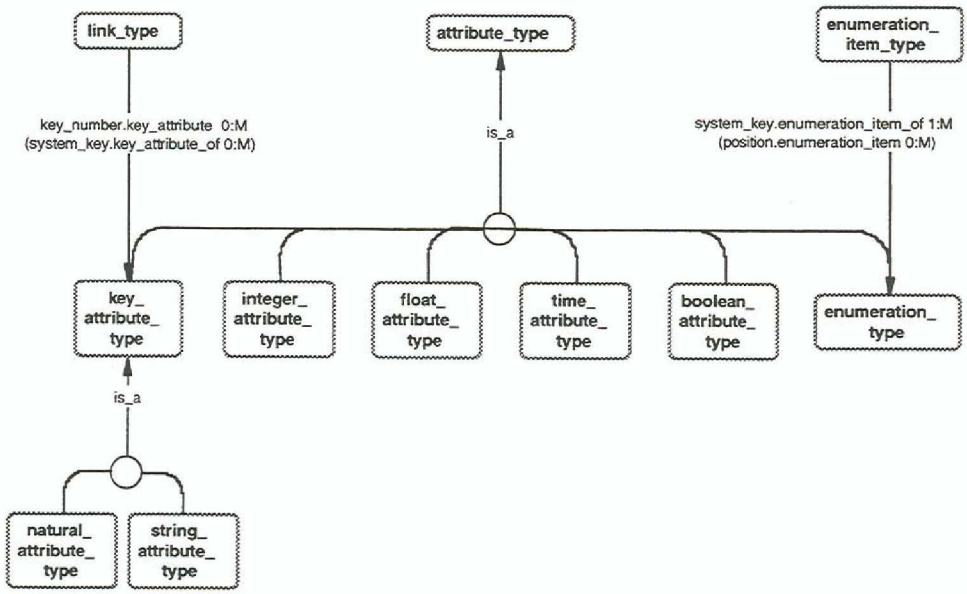
- Link_type är alltid av en viss kategori. Kategorin är intressant eftersom den bland annat reglerar vilka operationer som är tillåtna och under vilka förutsättningar. Här följer några kategorialternativ:

  – Composition link
  Destinationsobjektet hänger ihop med ursprungsobjektet som en komponent (uttrycker ett slags "part_of"-förhållande). Detta är bl a användbart i samband med låsningar, när man önskar skapa nya versioner av ursprungsobjekt (nya versioner av destinationsobjekten skapas automatiskt).

– Existence link
Tas en link bort försvinner även destinationsobjektet.

– Reference link
Destinationsobjektet får inte tas bort så länge dess link finns kvar.

Figur 19 visar den attribut-orienterade delen av metaschemat. Förutom många olika typer av attribute_types finns här också prefixet för en link_type definierat. Som namnet antyder fungerar det som en nyckel för link_type. M a o kan prefixet inte variera för varje SDS det används i.



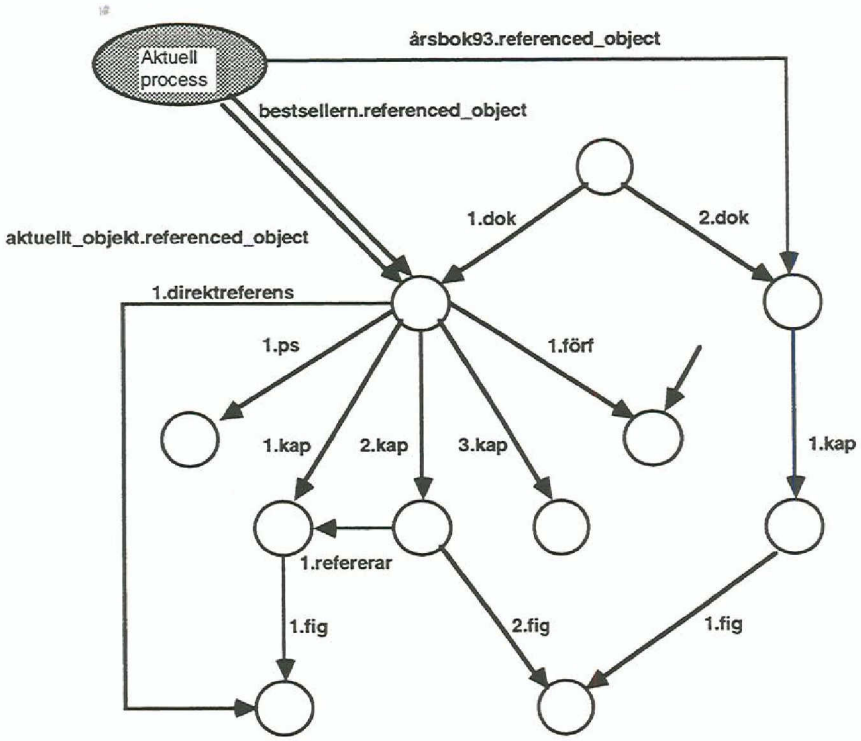
Figur 19

3.5 Principen för objektreferenser

Vi har redan konstaterat att både metaschema, schema och data ligger i samma resurskatalog och hanteras över samma gränssnitt, baserat på principerna för meta-schemat. De flesta operationer berör ett eller ett fåtal specifika objekt (på samma eller olika modellnivå) åt gången. Se exempelvis instruktionerna ovan, där tre eller fyra objekt är inblandade.

Hur refereras objekt i resurskatalogen på ett entydigt sätt? Till skillnad från i "vanliga" databaser har inte objekten någon nyckel, kombination av attribut etc, som kan användas. Visserligen har varje objekt en unik intern nyckel (surrogate, OID, internal, ...) men den är inte tillgänglig som en symbol att leverera över PCTE-gränssnittet. Grundprincipen i PCTE är istället att alltid ange en navigeringsväg från en given startpunkt över ett antal links fram till avsett objekt. Startpunkten kan bl a vara:

- det absoluta topp-objektet för PCTE-installationen, indikerat genom symbolen "_" (underscore)



Figur 21

Samtliga objektreferenser nedan pekar unikt ut samma objekt:

$/aktuellt_objekt/2.kap/2.fig

./2.kap/2.fig

$/årsbok93/1.kap/1.fig

_/execution_site_directory.system/0.known_execution_site/
0.initial_process/3.child_process/bestsellern.referenced_object/
2.kap/2.fig

under förutsättning att det sist berörda objektet dessförinnan är dokument 1 (1.dok) och att det sista exemplet är representativt för en navigering ända från topp-objektet.

3.6 Fördefinierad del av schemat

PCTE rekommenderar inget specifikt schema. Det är upp till varje verktygsleverantör att skapa ett sådant för sina behov. PCTE innehåller endast erforderliga mekanismer, enligt ovan, för att skapa helt eller delvis integrerade lösningar. Vissa undantag finns dock. Dels finns den statiska beskrivningen av aktuell PCTE-miljö i resurskatalogen tillsammans med dynamiskt tillstånd under körning (se vidare kapital 4). Dels finns en fördefinierad specialiseringshierarki av object_types, se figur 22. De flesta behövs för den egna statiska och dynamiska beskrivningen. Syftet är också att hierarkin ska fungera som en stabil startstruktur när egna scheman skapas.

- aktuell process, indikerad genom symbolen "$"
- aktuellt objekt (funnet genom en tidigare navigering), indikerat genom "."
- ett tidigare framnavigerat objekt, som givits ett unikt namn inom viss aktiv process (ungefär som en variabel).

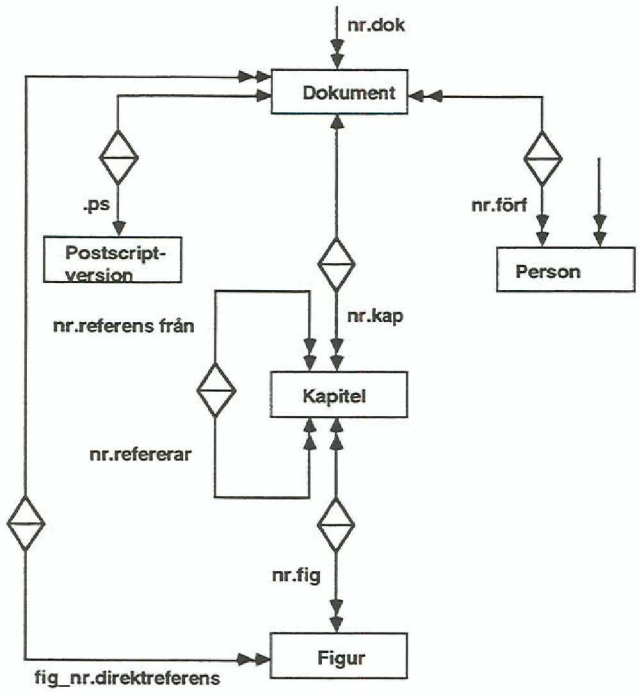
När ett startobjekt är givet sker sedan navigeringen över en eller flera link-förekomster. Varje link anges genom:

<key-value>.<link_type-name>

d v s ett exakt värde på prefixet i kombination med namnet på aktuell link-type. På så vis erhålls för det aktuella objektet en unik referens till en av dess links. Därmed är indirekt också destinationsobjektet för denna link given. Sedan är det bara att fortsätta därifrån med nästa steg i navigeringen enligt

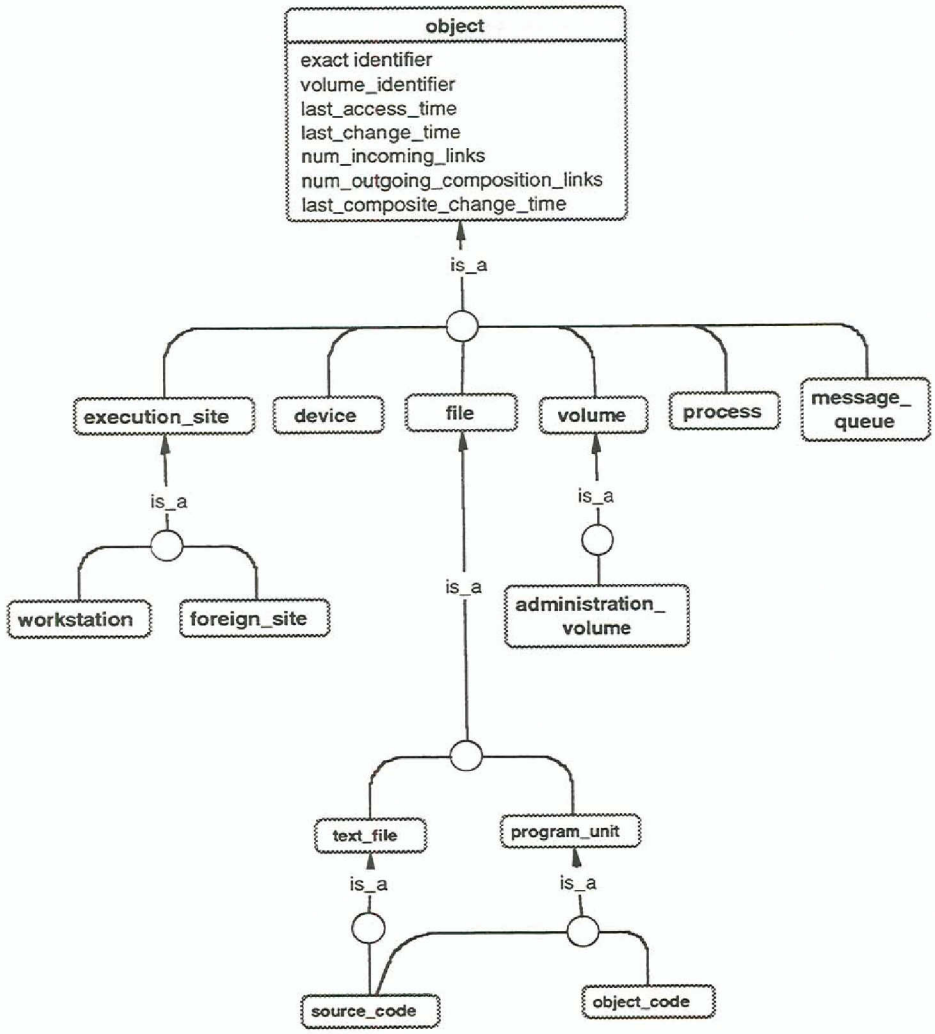
<referens till startobjekt>/<referens till linkförekomst>/ .....

Som synes ger en navigering alltid "träff" på högst ett objekt. Mängdoperationer som basfunktioner har inte bedömts vara erforderliga i PCTE-sammanhang.
Figur 20 visar ett schema och figur 21 en möjlig förekomstnivå.



Figur 20

Under körning placeras aktuell process som objekt i resurskatalogen. Från denna går bla links för att peka ut aktuellt objekt för respektive process samt aktuella objekt mot processens variabler (bestsellern, aktuellt_objekt, årsbok93).


object
exact identifier
volume_identifier
last_access_time
last_change_time
num_incoming_links
num_outgoing_composition_links
last_composite_change_time
is_a
execution_site
device
file
volume
process
message_ queue
is_a
is_a
workstation
foreign_site
is_a
administration_ volume
text_file
program_unit
is_a
is_a
source_code
object_code

Figur 22

4. Gränssnittsoperationer

Grundläggande operationer i PCTE är de som hanterar data mot resurskatalogen, t ex::

- att skapa nya objects och links eller att ge objects och eller links nya attributes
- att radera
- att ta kopior
- att söka

Motsvarande operationer behöver kunna appliceras på schemadata för hantering av scheman, t ex:

- att skapa nya object_types, link_types och attribute_types och deras samband inom givet SDS
- att radera
- att söka
- att importera

Instruktionerna i avsnitt 3.4 var exempel på den senare typen.

Data måste kunna hanteras i en distribuerad miljö, i fleranvändarmiljö, under lämplig behörighetskontroll och mot de villkor som formuleras i scheman (dataintegration). Verktyg som körs samtidigt behöver kunna styras och samordnas, de behöver kunna meddela sig med varann o s v (processintegration).

Förutsättningar och tillstånd för hantering av denna data/processintegration finns, som tidigare nämnts, beskrivna i resurskatalogen och de är strukturerade i enlighet med fördefinierade delscheman.

Hanteringen av dessa data sker med hjälp av en större uppsättning gränssnitts-operationer, grupperade per ändamål. Dit hör:

- hårdvarukonfiguration och hur data ligger distribuerade över konfigurationen
- behörighetsbeskrivning (för individer, grupper, program)
- versionshantering
- låsningshantering
- processhantering, -förutsättningar och -tillsånd
- meddelandehantering

Det skulle föra för långt att i denna översiktsrapport gå igenom alla delscheman och tillhörande gränssnittsoperationer (över 200 stycken). Som en exemplifiering av en av grupperingarna beskriver nästa avsnitt kortfattat hur låsning hanteras.

4.1 Exempel: Låsningshantering

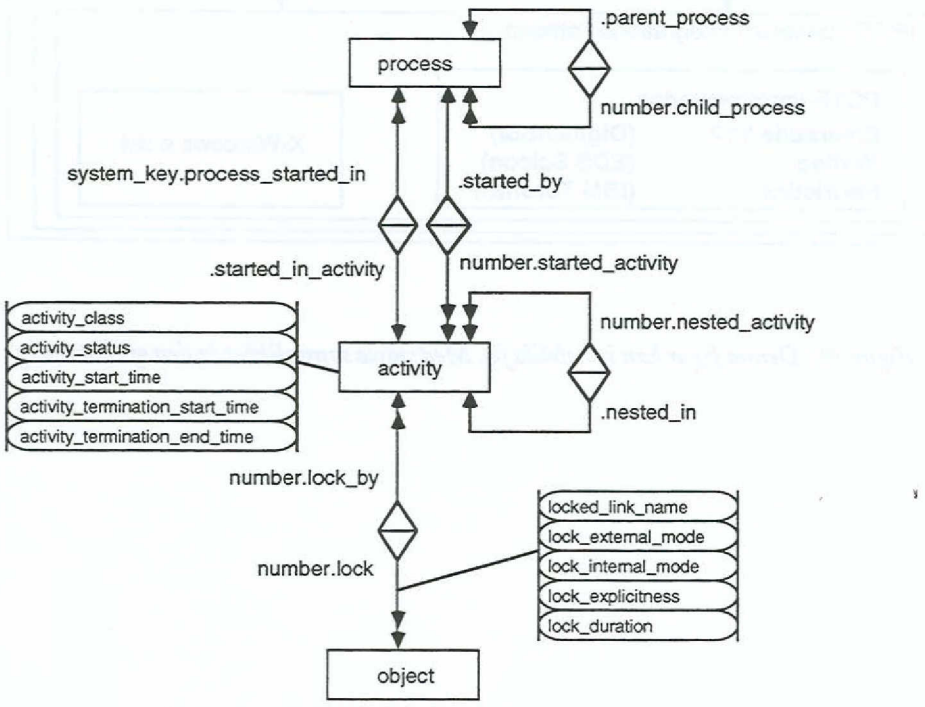
Låsning sker på objektnivån och begärs av en process. En komplicerande faktor är att man inom en viss process kan ha olika behov av låsning vid olika faser av processen. T ex: en viss fas inom en process kallas *activity*. Under en viss activitys levnad kan, förutom redan tidigare definierad låsning, tillfälligtvis andra kompletterande låsningar behövas. Dessa begärs i så fall från en annan activity. I det generella fallet kan denna överlappning (nestling) av activities bli obegränsat komplex. I en viss situation har PCTE att ta hänsyn till alla låsningar som begärts av alla, i ett visst läge aktiva, activities.

Inte blir det lättare av att processer kan anropa andra processer i en obegränsat komplex substruktur. Varje anropad process måste ta hänsyn till den anropande processens låsningskrav, förutom sina lokala.

En activity kan alltså sträcka sig över flera processer. Den refererar till de objekt som ska låsas och anger för var och en vilken typ av låsning (unprotected, semi-protected, protected, transactioned) som avses för vilken typ av operation (read, write, delete).

Transaktionshantering, exemplevis, ombesörjs genom en "transactioned" activity. Där tillkommer en komplikation, nämligen kravet på rollback om inte transaktionen som helhet kan genomföras korrekt. Eftersom activities kan vara överlappande betyder inte en bekräftelse (commit) av en activity otvetydigt en bekräftelse mot databasen. Finns en överordnad activity sker endast commit mot denna. Databasens innehåll påverkas definitivt först när översta activity kan göra sin commit. Begär den själv eller någon av dess underordnade activities en rollback måste en sådan göras över alla för tillfället inblandade activities.

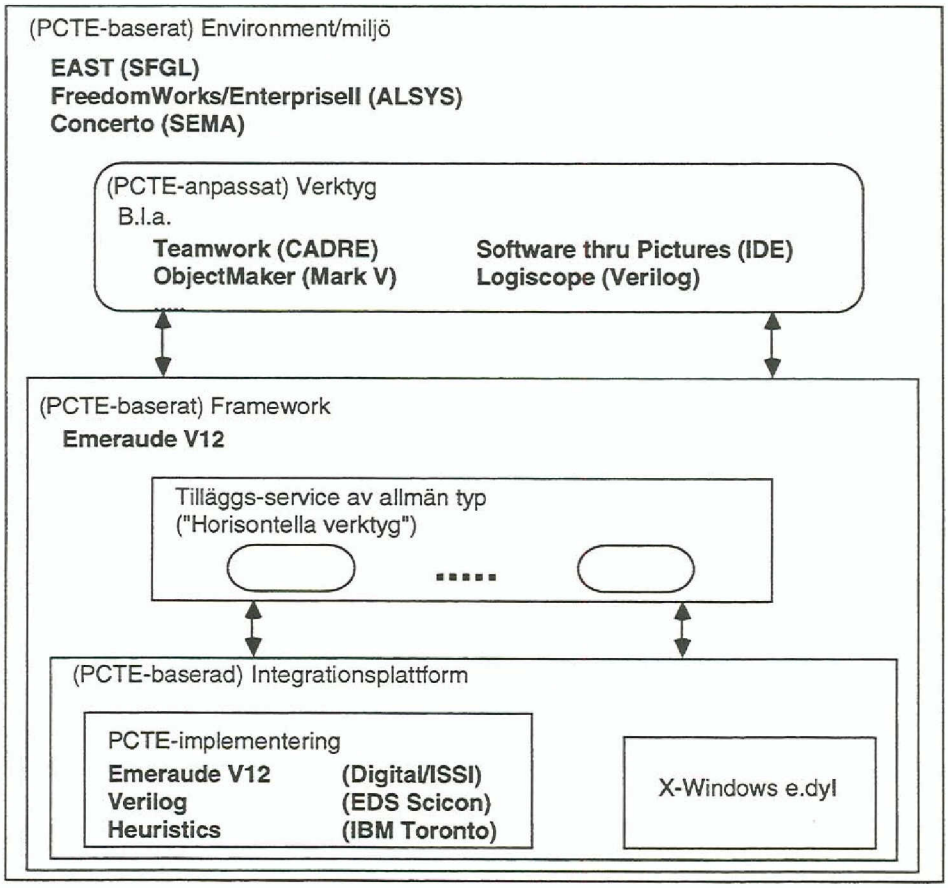
Data för denna komplicerade mekanism struktureras i enlighet med delschemat i figur 23. Lock_external_mode och lock_internal_mode för link_type number.lock anger typen av låsning. Låsningsprincipen kan skapa deadlocks. Det är upp till den som realiserar funktioner att att lösa upp dessa på ett korrekt sätt.



Figur 23

Övriga typer av gränssnittsoperationer är lösta med samma höga ambitionsnivå. Inte undra på att det tar ett antal år att nå en stabil, någotsånär felfri och heltäckande specifikation! Den intresserade kan själv botanisera inom de övriga områdena genom att läsa valda delar av ECMA 149.

5. Existerande produkter



Figur 24 (Denna figur kan innehålla fel. Med stösta sannolikhet är den ofullständig)

6. Stödorganisationer

Förutom arbetet inom ECMA TC33 finns ett flertal aktiva stödgrupper för PCTE.

- PIMB har ombildats till en ideell internationellt orienterad förening. Medlemsskap är öppet för företag mot en mindre avgift. Bla ger de ut ett newsletter som för närvarande distribueras fritt även till intresserade privatpersoner. Formellt heter organisationen nu PIMB Association.

För newsletter-prenumeration kontakta:

Emeraude
BP3, PC-7B11
Att: Ian Campbell
68 Route de Versailles
78430 Louveciennes
Frankrike
Tel: +33-1-3902-4001
Fax:+33-1-3902-4206
email: Ian.Campbell@frlv.bull.fr

För medlemskap kontakta:

DG XIII/A/4
European Commission
Att: Werner Wohlschlegel
Rue de la Loi, 200
B-1049 Brussels
Belgien
Tel: +32-2-296-8109
Fax: +32-2-296-8364

- NAPUG (North American PCTE User Group) är en påtryckningsgrupp om f n ca 170 medlemmar (personer).

- NAPI (North American PCTE Initiative) är en ny intressant gruppering bestående av:
  - NIST-CSL (NIST Computer Systems Laboratory)
  - DDI (Office of the Director of Defense Information)
  - OMG (Object Management Group)

NAPI lär ännu inte vara formellt etablerad. När så blir fallet blir de sannolikt en "maktfaktor" avseende PCTEs fortsatta öde.

Ytterligare en fråga för framtiden är om OMG:s medverkan kan indikera en framtida vinkling av specifikationen mot ett mer objektorienterat gränssnitt?

7. Några avslutande kommentarer

Till sist några kommentarer i punktform:

- PCTE-definitionen är stabil. Ca tio års ansträngningar ligger bakom definitionen och dessutom har många olika intressenter varit och är inblandade.

- Gränssnittet är mycket komplext. Så som det är uppbyggt idag kommer det att krävas en ny uppsättning operationer för varje tilläggsfunktion som kan komma att begäras framöver. Varför behövs t ex olika uppsättningar operationer för att hantera data respektive schemadata? De ligger ju i samma resurskatalog varför åtminstone viss överlappning borde vara möjlig.

- Fortfarande är PCTE tämligen oprövat. Visserligen kommer fler och fler produkter, men med ganska långa mellanrum. Beror det på tveksamhet kring grundidén eller komplexiteten?

- Europa tycks ha ett större och aktivare intresse än USA som tycks vara mer artigt intresserade. Kanske kommer detta att förändras när IBM:s AD-Platform realiseras med ett PCTE-gränssnitt (enligt ryktet).

- Fokus har legat på så kallade "coarse-grained objects", d v s objekt av storleksordningen fil och program. Jämför med PCTE:s grundläggande syfte. PCTE innehåller vissa enklare möjligheter att ta del av filers innehåll, men inte alls i enlighet med någon mer avancerad semantisk modell. Vissa hävdar att PCTE:s principer kan anpassas till hantering även av "fine-grained objects" utan problem medan andra hävdar att detta kommer att innebära stora prestandaproblem. Prestanda kan visa sig vara ett problem i alla händelser.

- Alla de mekanismer som finns definierade genom PCTE-gränssnittet i en integrationsplattform behövs säkerligen. Frågan är dock om inte dagens eller morgondagens avancerade databashanterare innehåller det mesta av dessa mekanismer i en generellare form. Av speciellt intresse är här mer objektorienterade databashanterare eftersom deras modelleringsbegrepp är ganska närliggande PCTE:s.

- Vissa modellegenheter, såsom prefixerade linknamn, fristående links, attribut på links istället för på sambandet i sin helhet o s v, känns som eftergifter mot realiseringsstrategier. Sammalunda gäller principerna för det navigerande sättet att referera till objekt.

- Frågan är också om den till synes finurliga principen om att lagra såväl data som schema i samma databas enbart är positiv? Den stämmer inte med de principer som finns redovisade i ISO/IEC JTC 1's Reference Model for Data Management, DIS 10032.

- Det spelar ingen roll hur fin en integrationsplattform är om det inte finns överenskomna scheman att arbeta efter. Det viktigaste och kanske svåraste jobbet ligger antagligen där. PCTE arbetar dock inte med denna frågeställning.

- Oklart är också viljan och förmågan till integrering av delvis överlappande SDS. Vem tar initiativ och vem tar ansvar? Vem kompromissar fram samordningen och hur ska det gå till?

- Det bör noteras att det finns konkurrerande ansatser. Främsta företrädare härvidlag är olika IRDS-standarder från ANSI och ISO. Objektorienterat angreppssätt vinner alltmer terräng. Vissa hävdar att komplexiteten skulle minska dramatiskt med en objektorienterad ansats.

- Till sist finns ju den allmänna debatten om och när mer eller mindre samordnade resurskataloger överhuvudtaget är realistiska alternativ.

Det är mycket som framtiden får utvisa.

Slutligen ett stort tack till Katarina Kindwall som så brutalt men mycket välmotiverat redigerat rapporten.

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 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

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.