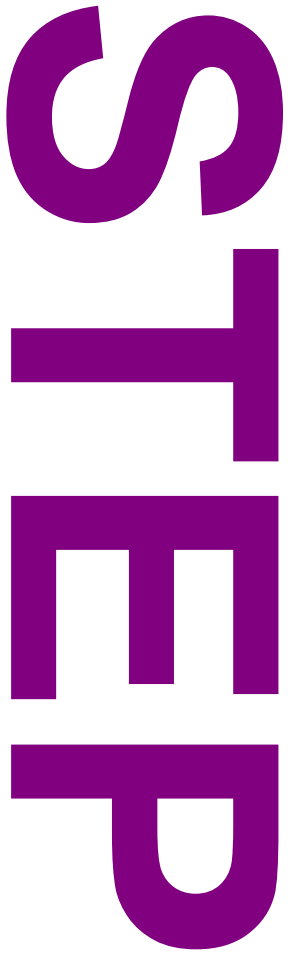
En skonsam men effektiv introduktion



Lars Celander

Om denna bok ...

I bästa Internet-stil, kan den här boken laddas ner gratis över nätet. Boken finns upplagd i form av en PDF-fil på adressen http://www.raserbaden.com.

Det är fritt fram att distribuera den vidare, både i elektronisk form och i pappersform. Dock skall boken alltid distribueras vidare i sin helhet och med copyright-notisen intakt.

Boken kommer att uppdateras kontinuerligt. Smärre justeringar och korrigeringar av i huvudsak redaktionell karaktär införs direkt, allt efter behov. Större innehållsmässiga uppdateringar kommer att genomföras varje år.

Bokens läsare är mycket välkomna att komma in med synpunkter på boken och med förslag på förbättringar av den. Det finns alltid utrymme för att göra den ännu bättre.

Lars Celander

E-post: lars@raserbaden.com

INNEHÅLL

Förord

Denna boken är tänkt som en grundläggande introduktion till STEP. STEP är en komplex standard. Att sätta sig in i vad STEP är, och inte är, kan vara besvärligt. Syftet med boken är att göra STEP mer lättbegripligt.

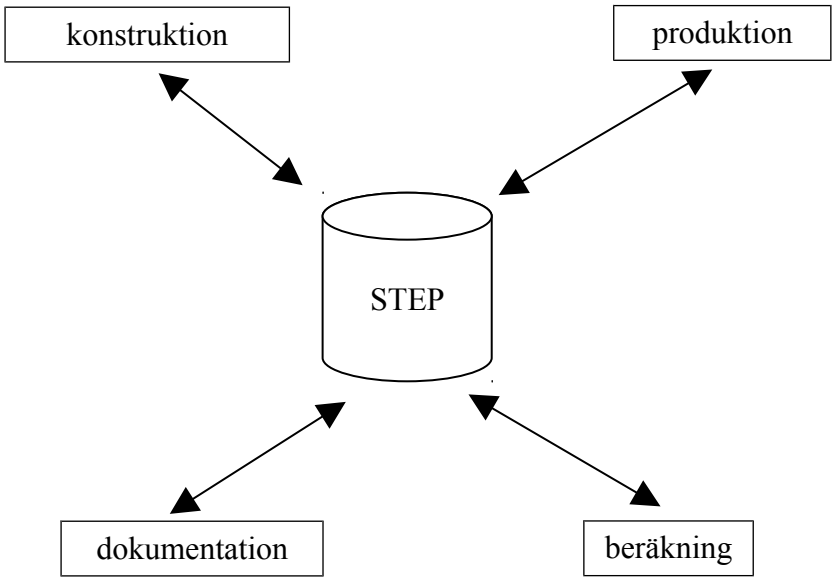
Avsikten är alltså inte att "sälja" STEP, dvs. att tala om hur fantastiskt det är och hur det kan lösa alla världens problem. Avsikten är bara att på ett kunnigt, nyanserat och pedagogiskt sätt, ge folk en god uppfattning om vad STEP i praktiken innebär.

1 Inledning

STEP är en internationell ISO-standard för olika former av produktbeskrivande data. STEP var ursprungligen inriktat på mekaniska detaljer men användningen har successivt vidgats till även omfatta elektronik (kretskort), elinstallationer, skeppsbyggnad, offshore, processindustri och byggnader.

Det grundläggande syftet med STEP-standarden är att göra det möjligt att flytta information mellan olika system. Alla företag måste inte ha samma system. Olika företag, eller olika delar av samma företag, kan samarbeta runt en produkt ändå. Syftet med en standard är också att långtidssäkra information, dagens system kan ju vara sällsynta om 50 år.

Även inom ett företag kan man ha olika system.



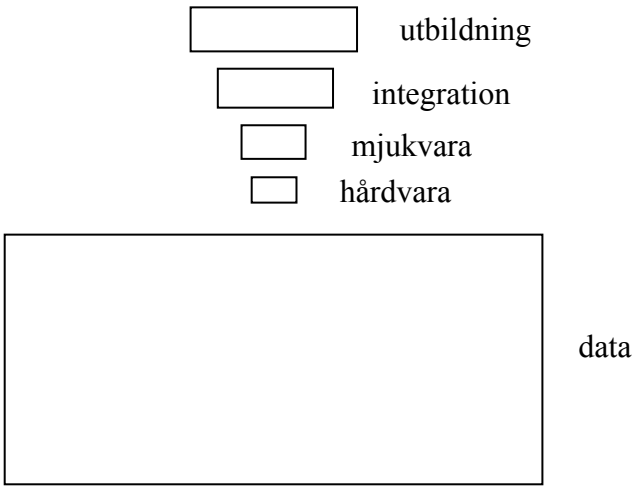
2 Ekonomiska drivkrafter

Kostnadsbilden

Industrin lägger ner oerhörda summor pengar på att beskriva och dokumentera sina produkter. Det rent praktiska och påtagliga resultatet av utvecklingsarbetet blir olika former av produktbeskrivande datamängder.

Utvecklingskostnaderna för produkterna tenderar att bli en allt större andel av produktens kostnad. Ett sätt att hålla nere utvecklingskostnaderna, är att struktur-rationalisera till större företagsenheter, något som görs intensivt i relativt mogna branscher som till exempel fordonstillverkning. På detta sätt kan utvecklings-kostnaderna slås ut på längre serier.

Ett annat sätt att angripa problemet, är att effektivisera utvecklingsprocessen. Det kan då vara intressant att studera hur kostnadsbilden ser ut, uttryckt i hur mycket pengar man lägger på olika aspekter av utvecklingsprocessen.



Figur 1 Typisk kostnadsbild

Egentligen är den här bilden rätt självklar. Den i särklass största utgiften är ju helt enkelt arbetstiden att ta fram underlag och dokumentation. Kringutgifterna för verktyg i form av mjukvaror och datorer, är i sammanhanget rätt obetydliga.

Det blir alltså i hanteringen av informationen som de stora produktivitetsvinsterna kan göras. Ett sätt att höja produktiviteten blir att återanvända information. För detta krävs standarder för hur informationen struktureras och representeras, vilket är vad STEP tillhandahåller.

Vad bilden säger är att IT egentligen handlar om informationsteknik, inte om datorteknik. Öppna system är nog bra men det är i öppen information som de stora pengarna ligger. Det enorma genomslaget som e-post och HTML-sidor har fått, visar på möjligheterna när man lätt kan flytta runt information.

Vill man måla i stora penseldrag, kan man se IT-branschen som bestående av i huvudsak tre sektorer, alla med olika mognadsgrad. Den första sektorn är själva datorerna. Den sektorn är nu relativt mogen. Datorerna är billiga och rätt väl standardiserade volymprodukter. IBM dominerade i början med sina stordatorer, senare Intel med högprestanda-PC och nu senast Microsoft med sina operativsystem och Office-sviter.

Den andra sektorn är kommunikation, där nu plötsligt TCP/IP genom Internet har blivit en de facto standard. Den branschen är i snabb förändring men kommer sannolikt att mogna rätt snabbt när bredbandsanslutning så småningom blir både vanligt och billigt.

Kvar som kostnadskritisk del blir den tredje sektorn: informationen. Den här sektorn är i många olika mognadsstadier samtidigt. Massmedia är en relativt mogen del där det är svårt att etablera en ny standard, till exempel digital-TV eller DAB. För de flesta tillämpningar är dock informationen en omogen sektor. Möjligheterna med Internet har nu satt igång en enormt snabb utveckling. De närmaste åren kommer att bli mycket spännande. HTML täcker in långt ifrån alla behoven. XML ser ut att bli den generella lösningen för att hantera mer sofistikerade behov. STEP är en utpräglad nischlösning men en tung och viktig sådan. Hur arbetsfördelningen mellan XML och STEP kommer att bli i framtiden, återstår fortfarande att se.

Slutresultatet för industrins IT-budgetar, blir att man i framtiden kommer att spendera mindre pengar på burkar och sladdar, i takt med att dessa blir allt billigare, och istället spendera allt mer på hantering och överföring av information.

Tidsbilden

Kostnadsbilden är inte det enda perspektivet. Med allt kortare produktcykler blir utvecklingstiden starkt kopplad till produktens lönsamhet, ibland helt avgörande för den totala lönsamheten.

Det finns flera vägar att korta utvecklingstiderna. De flesta handlar om att antingen minimera antalet process-steg eller att flera process-steg körs samtidigt ("Concurrent Engineering"). Dessa tenderar alla att kräva bättre kommunikation av produktdata.

Leverantörsrelationen

För att ta fram olika typer av beskrivningar och underlag, används idag nästan uteslutande olika former av datorbaserade verktyg, till exempel MCAD-, ECAD-, eller GIS-verktyg.

De leverantörer som tillhandahåller dessa verktyg har inget intresse av att kunderna lätt skall kunna byta till något konkurrerande verktyg. Man har heller inget intresse av att kunden skall kunna köpa tilläggsfunktionalitet från vem som helst. Givetvis vill man styra kunden till att köpa de tilläggsmoduler som man själv

tillhandahåller. Dessa intressen bevakar man genom att styra användarens tillgång till de data som ligger i verktyget. Är användarens data inlåst, är användaren inlåst. Ett tydligt exempel är hur Autodesk hemlighåller specifikationen på sitt DWG-format, allt för att försvåra för konkurrenterna. Å andra sidan vill självklart användaren har fri tillgång till informationen för att kunna bygga upp de informationsflöden som är optimala för verksamheten.

Det pågår alltså en dragkamp mellan leverantören och användaren om vem som har makten över informationen som ligger i verktyget.

Det intressanta här är, att som regel är de vinster som leverantören kan göra genom inlåsning och genom minskad konkurrens mycket mindre, än de förluster som användaren gör på grund av svårigheter i att flytta information. Det finns alltså ett stort utrymme för ömsesidigt lönsamt samarbete mellan leverantör och användare, något som man också ofta ser i form av olika allianser och samarbetsprojekt. Det är dock bara de riktigt stora företagen som har tillräckligt med tyngd för driva en sådan utvecklingslinje.

I det här spelet intar STEP en rätt tuff hållning. Syftet med STEP är uttryckligen att helt frikoppla data från verktyget och SDAI (se senare) gör detta fullt möjligt. I princip kan ett verktyg reduceras till en editor för STEP-data, lätt att snabbt byta ut när något billigare eller bättre dyker upp på marknaden. Det är fler än en marknadsansvarig på något leverantörsföretag, som har blivit lite blek om nosen när de har förstått slutmålet med STEP. Här går STEP en balansgång. Det gäller ju att få med sig leverantörerna också.

Produktiviteten i en fristående arbetsplats

De stora produktivitetshöjningarna tycks här ligga i att man över huvud taget använder datorbaserade hjälpmedel och att man gör det på ett smart sätt. Produktiviteten verkar mycket att hänga ihop med matchningen mellan verktyget och arbetsmetoden. Bilden är rätt splittrad och det är inte lätt att se några enkla och snabba vägar till generellt högre produktivitet.

Man skulle annars kunna tänka sig att göra rent STEP-baserade verktyg, alltså verktyg som har STEP som internt format. Troligen är detta fel väg att gå. Låsningen till STEP innebär ofta begränsningar för verktygen, vilket genom att utvecklingen stoppas upp, riskerar att gå ut över produktiviteten. Rent psykologiskt är inte heller hårt standardiserade STEP-baserade verktyg alltid så charmiga. Folk vill ha sina finesser och leverantörerna tillhandahåller dem villigt. Standardisering må vara lönsamt men det är helt enkelt inte roligt. Den IT-chef som slår ner på teknikglädjen blir lätt impopulär.

Kostnaden för att återanvända data

Att överföra information må vara lönsamt, men det är inte gratis. Det finns i regel en fast kostnad associerad med att göra överföringen möjlig och tillförlitlig. När man

överför mycket komplex information, typ den som STEP hanterar, kan det lätt röra sig om ett antal manmånader, även om kostnaden kan variera stort från fall till fall. Den rörliga kostnaden, arbetet för att göra en viss överföring, är också mycket beroende på omständigheterna. Någon arbetstimma är inget ovanligt.

Oavsett hur stor kostnaden att använda STEP är, så finns den där. Finns det en färdig integration mellan två verktyg eller verktygsdelar, är det givetvis effektivare att hoppa över STEP-steget.

Produktiviteten i hela utvecklingsprocessen

Utvecklingsprocessen ses här i vid mening, alltså även inklusive dokumenterande och stödjande funktioner under produktens hela livscykel.

För tätt kopplade lokala processer använder man troligen en uppsättning väl integrerade verktyg, troligen från en enda leverantör.

För lösare kopplingar mellan processer på högre nivå, är troligen STEP ett starkt alternativ. Överföringarna görs mera sällan och en storskalig samordning till en enda leverantör är knappast önskvärd.

3 Sponsorer och organisation

STEP drivs fram av användare, i stort sett på frivillig basis. STEP-arbetet har ingen egen finansiering utan är helt beroende av att stora företag låter anställda lägga tid på det eller på statlig finansiering via olika projekt, institut och organisationer. Avsaknaden av egen finansiering inom STEP gör att projektstyrningen blir rätt svag. Arbetet beror helt på vad folk är intresserade av och har finansiering för.

STEP-möten brukar bevistas av ett hundratal deltagare. Kärnan är sedan lång tid etablerad och där är omsättningen på folk lyckligtvis förvånansvärt låg. STEP-arbetet lider annars kronisk brist på kompetent och finansierat folk, i synnerhet i förhållande till komplexiteten i vad man försöker åstadkomma. Arbetet tenderar att dra ut på tiden. Det är ändå mycket imponerande vad som har åstadkommits.

Leverantörerna av olika former av system har, som tidigare nämnts, en klart ambivalent inställning till STEP. Rent kommersiellt har man inget större intresse av STEP, i stora delar är ju STEP ett direkt angrepp på deras möjligheter att låsa in kunden i proprietära format. Å andra sidan kräver kunderna uttryckligen stöd för STEP så något måste de ju göra. En fördel med STEP, ur leverantörernas synvinkel, är att användarna har enats om en enda standard, istället för att varje grupp av användare skall ha stöd för just sin "standard".

Organisatoriskt sett ligger STEP-arbetet under ISO (International Standards Organisation). ISO har ett antal Technical Committees (TC) varav STEP sorterar under TC 184 Industrial Automation Systems and Integration. Denna kommitté är i sin tur uppdelad i olika Sub-Committees varav STEP ligger i SC4 Industrial Data. Under SC4 ligger sedan ett antal arbetsgrupper (Workgroups, WG) där det praktiska arbetet bedrivs. SC4 sköter inte bara STEP (ISO 10303) utan även sådant som ISO 13584 Parts Library.

Arbetet i SC4 styrs av en styrkommitté i vilket ett antal länder, däribland Sverige, har valt att deltaga med en representant. Dessa röstar om stora och små beslut i hur STEP-arbetet bedrivs. Godkännandet av standarder, däribland givetvis STEP:s olika delstandarder, godkänns i omröstning med samtliga ISO-länder, alltså inte bara de länder som sitter med i SC4.

I Sverige finns SMS-kommitté 2169 där det sitter representanter från svensk verkstadsindustri. Huvudsakliga sysselsättningen är att vara ett formellt organ för att bestämma hur Sverige skall rösta i SC4 och om olika delstandarder.

STEP heter egentligen ISO 10303. STEP förekommer aldrig officiellt i ISO-dokumenten men är det namn alla använder. STEP består av många delstandarder. Dessa ges ett löpnummersuffix enligt ISO 10303 - nn.

ISO har en systerorganisation som heter IEC (International Electrotechnical Organisation). Arbetsfördelningen är att ISO har hand om allt "icke-elektriskt" och IEC har hand om allt "elektriskt". Man strävar att samarbeta så mycket som möjligt men uppdelningen gör ändå att det tenderar att bli två olika världar. Problemet är tydligt för till exempel kretskort eller elinstallationer. En standards genomslag handlar mycket om "mindshare". STEP är inte väl inarbetat bland "elektriker" och man har en tendens att se STEP som ytterligare en standard i en redan överbefolkad och illa integrerad flora av standarder.

Utvecklandet av nya ISO-standarder följer en noga definierad process. Processen tar lång tid, på gott och ont. Fördelarna är att det ger tid till förankring samt en god stabilitet, bägge faktorerna är viktiga för en standard. Nackdelen är självklart att arbetet

går trögt. Processen är uppdelad i ett antal stadier. En ny standard har olika benämningar beroende på i vilket stadium den befinner sig:

1. NWI (New Work Item Proposal)) innebär att någon föreslår att något skall göras. I STEP-sammanhang är det SC4 som beslutar om officiellt arbete skall börja.

2. WD (Working Draft) innebär att arbete pågår. Ett WD är troligen varken stabilt eller komplett.

3. CD (Committee Draft) innebär att arbetet har resulterat i något som ansvarig arbetsgrupp anser vara relativt klart och moget för en större publik. Börjar nu cirkuleras officiellt till nationella standardiseringsmyndigheter. En ny standard kan ligga på denna nivån rätt länge och ibland kan avsevärda förändringar göras under perioden. Mot slutet av CD-stadiet kan man börja betrakta standarden som stabil.

4. DIS (Draft International Standard) innebär att konsensus har nåtts. ISO har särkilda definitioner på vad som menas med "konsensus".

5. FDIS (Final Draft International Standard) innebär att den blivit godkänd i en första ISO-omröstning.

6. ISO (International Standard) innebär att den är godkänd i slutlig ISO-omröstning och därmed alltså så klar den kan bli.

Ett dokument ges ibland status som Technical Specification (TS). TS innebär i princip ingenting. Det kan möjligen betraktas som ett förstadium eller ett alternativ till WD.

ISO-arbetet är i dagsläget pappersbaserat. Dokument på elektronisk form är ännu inte fullt accepterade, vilket ställer till en del problem för STEP-arbetet. ISO har också copyright på alla DIS och IS-dokument och tar betalt för kopior. Försäljning sker genom nationella standardiseringsmyndigheter, i Sverige alltså genom SMS (se Bibliografin). Man kan tycka vad man vill om hur ISO fungerar, men det är så det fungerar. ISO-vägen är rätt väg att gå i många sammanhang men kanske inte i alla. Ibland är det bättre att gå förbi ISO.

4 Historisk bakgrund

Att flytta data mellan system har man hållit på med länge, med blandad framgång. Den klassiska metoden är att definiera ett standardiserat filformat.

Inom MCAD-världen är IGES (Initial Graphics Exchange Standard) en av de första och kanske den mest kända föregångaren till STEP. Enligt legenden snickrades IGES ihop en helg i en sportstuga i nordöstra USA, över ett antal öl. Att döma av hur standarden är skriven, är detta fullt möjligt. IGES är i grunden ett klassiskt amerikansk "quick-n-dirty" som sedan blivit de facto världsstandard. IGES klarar av det mesta, både i form av 2D ritningar och 3D modeller, i alla fall i teorin. I praktiken har man fått begränsa sig rätt hårt för att få en god tillförlitlighet i överföringarna.

Tyska bilindustrin har en lång historia av att energiskt ägna sig åt problemet. VDAFS (VDA Flächen-Schnittstelle) var en tidig standard som bara kunde skicka 3D ytor, men som gjorde det bra, kanske mest på grund av sin enkelhet. VDAIS (VDA Iges Subset) var ett försök att få ordning på IGES. Försöket var mycket lovvärt, men det fick tyvärr inget större genomslag i form av stöd från leverantörerna.

Fransmännen har naturligtvis sin egen standard kallad SET (Standard d'Echange et de Transfer), populär i Frankrike men ignorerad i resten av världen. Ungefär samma användningsområde som IGES och enligt uppgift med bättre tillförlitlighet.

Dessa tidigare försök har allesammans problem med att de inte kunde överföra all information man önskade sig. Man hade också problem med den tillförlitlighet man fick i överföringarna. Splittringen i flera överlappande standarder orsakar problem och merkostnader.

Efter att man hade lärt sig hur man skulle använda standarderna och efter att man blivit av med rena buggar, kvarstod ett mera fundamentalt problem. Problemet är strukturella skillnader mellan olika system, det faktum att systemen ofta valde helt olika begreppsuppsättningar när de konstruerades. Implicit i standarderna ligger också olika begreppsuppsättningar, vilket ytterligare komplicerar bilden.

Begreppet linjebredd är ett bra exempel. Följande användningssätt finns i användning, kanske finns det ännu fler varianter.

- linje med attributet REAL
- linje med fasta alternativ av REAL
- linje med INTEGER andel av REAL
- linje med attributet PEN
- linje med attributet FONT
- begreppet saknas
- begreppet simuleras med parallella linjer

Att översätta från en linjebredd uttryckt i en begreppsvärld till samma linjebredd uttryckt i en annan begreppsvärld, är inte på något sätt ett enkelt problem. I många fall får man helt enkelt göra ett allmänt vettigt antagande och hoppas på att användaren blir

nöjd. Ibland fungerar det, ibland inte. Grundproblemet är alltså olika och luddigt definierade begreppsvärldar.

Med STEP ville man skapa en enda internationell, enhetlig, integrerad, kraftfull och tillförlitlig standard. Arbetet började i mitten av 80-talet. Första delen av STEP blev klar i februari 1993. Fler delar har blivit klara sedan dess men mycket återstår fortfarande att göra.

En av grundidéerna bakom STEP är att explicit och entydigt definiera de begreppsvärldar som används i STEP. Detta görs i ett särskilt språk för informationsmodellering.

En annan av grundidéerna är att dela upp STEP i moduler allt efter applikationsområde, s.k. Applikationsprotokoll (se nedan).

Den tredje grundidéen var att skilja ut filformatet från resten av standarden. Filformatet ses nu som bara en av flera möjliga implementationsformer och kan enkelt ändras eller bytas ut om så behövs.

På sin tid var detta mycket avancerade idéer. De har stått sig bra och STEP betraktas idag som stilbildande på hur man bygger upp standarder av den här typen.

5 Teknisk beskrivning

Grundläggande principer

Det grundläggande syftet med STEP är att skapa standardiserade samlingar av begrepp. Kan man komma överens om begreppen, kan man också skicka information mellan varandra.

Dessa begreppsuppsättningar eller "språk", kallas med en fackterm för informations-modeller. Informationsmodeller beskriver alltså vilka begrepp som finns, vad de betyder och hur de hänger samman.

Inom STEP är informationsmodellerna grupperade i vad som kallas för Applikations-protokoll (Application Protocol, AP). Ett AP innehåller alltså en uppsättning begrepp. Vilka AP som finns definierade, och som stöds av leverantörerna, måste en användare känna till.

Inom STEP använder man ett av STEP utvecklat språk för informations-modellering, kallat EXPRESS. Alla AP använder EXPRESS. Rent principiellt är det inget som hindrar att något AP använder något annat språk men än så länge används uteslutande EXPRESS. Vilket språk som används behöver inte en normal användare bry sig om.

Ett AP är fördefinierat och i princip oföränderligt. Det innehåller vad det innehåller, på gott och ont. Man kan inte ändra i det för att det skall passa en själv bättre. Å andra sidan, man kan förvänta sig att andra, i princip vem-som-helst när-som-helst, skall vara införstådd med och kunna använda samma AP.

Ett AP är inget filformat. Information i enlighet med ett visst AP, kan överföras på många olika sätt. STEP definierar för närvarande två olika sätt att implementera själva den praktiska överföringen, dels via ett filformat och dels via ett programmerings-gränssnitt. Mer om dessa längre fram i texten.

Att använda STEP blir alltså att välja en kombination av ett AP och en implementations-metod.

Anatomin hos STEP

STEP är en komplex standard. Av begripliga skäl är STEP uppdelat i många delstandarder. Långt ifrån alla är klara. En stor del av STEP är alltså fortfarande under utveckling.

Alla delar av STEP har ett nummer. Tabellen nedan listar (i nummerordning) de nummerserier som är definierade. Nummerserierna är inte kristallklart logiskt disponerade, mycket av historiska skäl.

1	Overview and fundamental principles
11+	Description methods
21+	Implementation methods

31+	Conformance testing
41+	Integrated generic resources
101+	Integrated application resources
201+	Application Protocols (AP)
301+	Abstract Test Suites (ATS)
501+	Application Interpreted Constructs (AIC)
1001+	Application Modules (AM)

Applikationsprotokoll (AP)

Sett ur en användares synvinkel, består STEP av ett antal AP. Ett AP täcker in ett visst applikationsområde, till exempel plåtbockning (AP 207) eller rördragning i ett fartyg (AP 217).

Det finns ingen hierarki av AP, alla AP är jämlika. Inget sägs om hur man som användare hanterar överlapp mellan AP eller luckor mellan AP.

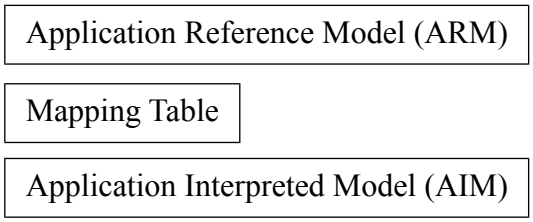
En implementation av STEP är alltid en implementation av ett visst AP. Man kan alltså inte bara hänvisa till att man använder STEP, man måste alltid tala om vilket AP.

Att sätta sig ner och börja läsa ett AP, kräver rätt starka nerver. Det brukar handla om en kraftig bunt av inte särskilt inbjudande läsning. Grundstrukturen är ändå rätt enkel. För fullständighetens och tydlighetens skull finns det en hel del redundans i ett AP-dokument, så innehållet är inte så omfattande som man först kanske tror.

Ett AP består i princip av en lång lista på vilka begrepp som används. Dessa begrepp skall vara de begrepp som är intressanta för applikationsområdet och uttryckta i applikationsområdets terminologi. Begreppslistan kallas för Application Reference Model (ARM). Sättet att exakt definiera begreppen kan variera men vanlig prosa och EXPRESS är de mest populära sätten.

I princip hade man kunnat stanna här. För att få bättre integration mellan olika AP, väljer man att gå ett steg till. Man kräver att dessa begrepp skall översättas till de begrepp som finns tillgängliga i STEP:s standardmodeller (40/100/500-serierna, se nedan). I en lång lista talar man om vilka begrepp man har valt att använda. Denna lista kallas för Application Interpreted Model (AIM). Syntaxen för AIM är alltid EXPRESS.

Översättningen från ARM till AIM definieras i en (mycket) lång lista kallad Mapping Table. Tabellen består i princip av ARM-begreppen i en kolumn och motsvarande AIM-begrepp i andra kolumnen. Vissa skrivkonventioner används men någon formell syntax är inte definierad. En Mapping Table är alltså inte tolkbar för en dator (se dock om EXPRESS-X längre fram).



Figur 2 Grundstrukturen hos ett applikationsprotokoll (AP)

Vissa AP kräver inte bara starka nerver, utan även starka muskler. Med ett omfång på flera tusen sidor börjar det fysiska omfånget bli ett problem. AP 210 på 3700 sidor är inget man stoppar ner i väskan för att sitta och bläddra i på flyget. Man får helt enkelt inte. Med en pappersvikt på 80 g/kvm blir vikten 18,5 kg, väl över gränsen för handbagage. Alla AP-dokument följer en viss given mall:

Kapitel	1	Syfte
	2	Normativa referenser
	3	Definitioner
	4	ARM
	5	Mapping Table

Appendix	A	AIM (fullständig listning i EXPRESS)
	B	AIM förkortningar (för implementationer)
	C	Formulär för konformitetstestning ("PICS")
	D	Implementations-schema
	E	Aktivitetsmodell ("AAM")
	F	ARM på grafisk form (t.ex. EXPRESS-G)
	G	AIM på grafisk form (t.ex. EXPRESS-G)
	H	Tillämpningsguide
	J	Teknisk diskussion och förklaringar
	K	Bibliografi

Som synes finns det en hel del i ett AP-dokument som är där av andra skäl än strikt definitionsmässiga. Beroende på vad man är ute efter, kan man dyka ner i den delen som bäst förklarar vad man är ute efter att få reda på.

Till sist, det kan vara värt att påpeka att det är AIM som används för implementationer. Man skulle kunna använda ARM (förutsatt att den är definierad i EXPRESS), debatten har förts livligt, men man har i slutändan valt att använda AIM och enbart AIM.

AP färdiga som ISO-standarder (IS)

AP201 Explicit draughting

Stödjer 2D ritning utan koppling till 3D modell.

AP202 Associative draughting

Stödjer 2D ritning med koppling till 3D modell.

AP203 Configuration controlled design

Stödjer 3D modell med konfigurationsinformation. Ingen 2D ritning. Det mest kommersiellt etablerade av alla AP.

AP207 Sheet metal die planning and design

Avsett för beskrivning av en plåtdetalj och det verktyg som pressar detaljen, inklusive relationen mellan plåtdetaljen och pressverktyget.

AP210 Electronic assembly, interconnect and packaging design

Beskriver kretskort. Stödjer en sammanhållen beskrivning av kretskortet ur både mekanisk och elektrisk funktionell synvinkel. Man slipper alltså, som nu är fallet, att blanda olika standarder för olika aspekter av kretskortet. Angreppssättet är unikt, mycket kraftfullt och rymmer stora möjligheter för elektronikindustrin. Ambitionsnivån innebär samtidigt att detta är det mest komplexa av alla AP med bortåt 4000 sidor.

AP212 Electrotechnical design and installation

För funktionell beskrivning av elektriska installationer, till exempel i en bil eller i ett kärnkraftverk.

AP214 Core data for automotive mechanical design processes

Bilindustrins eget AP. Filosofin är ett branschspecifikt AP, inte ett teknikspecifikt AP som de flesta andra AP. Resultatet blir ett mycket stort och omfattande AP men också ett väl integrerat hanterande av flera olika applikationsområden. Med bilbranschens relativt stabila behovsbild, är detta troligen den optimala vägen att gå. En avgörande aspekt är att AP214 har ett mycket gott stöd från leverantörerna. Noteras bör att AP214 är tänkt att användas ihop med AP212

(Electrotechnical Design and Installation), vad gäller fordonets elsystem.

AP224 Mechanical parts definition for process planning using machining features

Avsett för beskrivning av mekaniska detaljer tillverkade genom avverkande bearbetning (fräsning eller svarvning). Uppbyggt runt användningen av formelement ("form features"). Stödjer beredning, tillverkning och produktionsplanering men är inte tänkt att användas i konstruktionsfasen.

AP225 Building elements using explicit shape representation

Beskriver stommen på en byggnad i form av en 3D modell. Stödjer begreppen väggar, golv, bjälkar, grund, fönster och dörrar. I övrigt ingen intelligens utöver detta.

AP227 Plant spatial configuration

För rumslig beskrivning av en industriell anläggning, både för tillverkningsindustrin och för processindustrin.

AP på FDIS-stadiet

AP209 Composite and metallic structural analysis and related design

Avsett för beskrivning av detaljer uppbyggda av kompositmaterial. Stödjer för typen speciella data som skikt, skiktgränser och fiberriktning. Avsedd att användas för konstruktion och för olika typer av beräkningar.

AP på DIS-stadiet

AP204 Mechanical design using boundary representations

Redundant AP, kommer inte att utvecklas vidare till.

AP213 Numerical control process plans for machined parts

För processplanering av tillverkning av mekaniska detaljer via avverkande bearbetning. Stödjer relationer mellan produkter, processer och resurser.

AP232 Technical data packaging: core information and exchange

För ett konfigurationsstyrt utbyte av en sammanhållen grupp av produktrelaterad information, typiskt mellan PDM-system. Avsedd att användas för alla typer av produkter.

AP på CD-stadiet

AP216 Ship moulded forms

Ett fartygs skrovform, inklusive viktigare mellanväggar, i olika representationer. Inkluderar också hydrostatisk analys.

AP221 Functional data and their schematic representation for process plant

För funktionell beskrivning av en industriell anläggning, både för tillverkningsindustrin och för processindustrin.

AP231 Process engineering data: process design and process specification of major equipment

För process-konstruktion, process-simuleringar och konceptuell utrustnings-konstruktion. Primärt avsedd för kemisk industri och för olje- och gas-anläggningar.

AP ännu ej på CD-stadiet

AP215 Ship arrangement
AP217 Ship piping
AP218 Ship structures
AP220 Process planning for manufacturing and assembly of layered electrical products
AP223 Exchange of design and manufacturing product information for casting parts
AP226 Ship mechanical systems

AP230	Building structural frame: steelwork
AP233	Systems engineering data representation
AP234	Ship operational logs, records and messages
AP235	Materials information for design and verification of products
AP236	Furniture product and project data

Modularisering av AP

Med rätt många AP med snarlika innehåll och med akuta problem i omfånget hos olika AP, börjar det bli vettigt att försöka införa någon form av modul-system för hur man bygger upp ett AP. Modulariseringen har nyligen påbörjats och är alltså ännu långt från genomförd.

Man inför en typ av moduler som man kallar för Application Modules (AM). Tanken är att man relativt snabbt och enkelt skall kunna bygga upp ett AP bara genom att peka ut några lämpliga applikationsmoduler. Syftet med modulerna är alltså i huvudsak att förenkla STEPs interna dokumenthantering.

Modulera är kompletta små mini-AP i den meningen att de har både ARM och AIM och mappningen mellan dessa. Modulerna kan anropa varandra och kommer alltså att bilda en hierarki. Högst upp ligger relativt applikationsnära moduler och längre ner ligger enklare och mer generellt användbara moduler. Allra längst ner i modul-hierarkin sker anrop till resursmodellerna (se nedan).

Applikationsmodulerna är inte ISO-standarder i formell mening, de hanteras som Technical Specifications.

Part 1001	Appearance assignment
Part 1002	Colour
Part 1003	Curve appearance
Part 1004	Elemental shape
Part 1005	Elemental topological shape
Part 1006	Foundation representation
Part 1007	General surface appearance
Part 1008	Layer assignment
Part 1009	Shape appearance and layers

Resursmodeller

Applikationsprotokollen byggs upp med hjälp av färdiga modellsnuttar. Fördelen är dels att det går fortare, dels att det ger en viss integration mellan olika AP. Allmänt användbara modeller ligger i 40-serien ("Integrated generic resources"):

Part 41 (IS)	Fundamentals of product description and support
Part 42 (IS)	Geometric and topological representation

Part 43 (IS)	Representation structures
Part 44 (IS)	Product structure configuration
Part 45 (IS)	Materials
Part 46 (IS)	Visual presentation
Part 47 (IS)	Shape variation tolerances
Part 49 (IS)	Process structure and properties
Part 50 (DIS)	Mathematical constructs
Part 51 (CD)	Mathematical description

Mer applikationsorienterade modeller ligger i 100-serien ("Integrated application resources"):

Part 101 (IS)	Draughting
Part 104 (IS)	Finite element analysis
Part 105 (IS)	Kinematics
Part 106 (WD)	Building core model
Part 107 (CD)	Engineering analysis core application reference model
Part 108 (WD)	Paramerisation and constraints for explicit geometric product models
Part 109 (AWI)	STEP assembly model for products

För att kunna gruppera grundmodellerna i 40/100-serierna, i mer hanterbara och något mer applikationsorienterade block, skapade man 500-serien (Application Integrated Constructs, AIC):

Part 501 (IS)	Edge-based wireframe
Part 502 (IS)	Shell-based wireframe
Part 503 (IS)	Geometrically bounded 2D wireframe
Part 504 (IS)	Draughting annotation
Part 505 (IS)	Drawing structure and administration
Part 506 (IS)	Draughting elements
Part 507 (IS)	Geometrically bounded surface
Part 508 (IS)	Non-manifold surface
Part 509 (IS)	Manifold surface
Part 510 (IS)	Geometrically bounded wireframe
Part 511 (IS)	Topologically bounded surface
Part 512 (IS)	Faceted boundary representation
Part 513 (IS)	Elementary boundary representation
Part 514 (IS)	Advanced boundary representation
Part 515 (IS)	Constructive solid geometry
Part 517 (IS)	Mechanical design geometric presentation
Part 518 (DIS)	Mechanical design shaded presentation
Part 519 (IS)	Geometric tolerances
Part 520 (IS)	Associative draughting elements

Informationsmodelleringsspråket EXPRESS

Informationsmodeller skrivs i språket EXPRESS. EXPRESS ser ut ungefär som vilket programmeringsspråk (typ Pascal, C/C++, Java) som helst. EXPRESS är mycket starkt på typdeklarationer. Algoritmer och flödeskontroll ser ut som det brukar göra. Man skulle i princip kunna kompilera och köra EXPRESS-kod men det är det ingen som gör. EXPRESS saknar till exempel helt funktioner för I/O. EXPRESS beskrivs i en delstandard:

Part 11 (IS) EXPRESS language reference manual
Part 12 (IS) EXPRESS-I language reference manual

Part 12 EXPRESS-I definierar en variant av EXPRESS som används för att skriva testfall (se nedan om konformitetstestning).

EXPRESS-modeller kan inte innehålla svenska tecken. Teckentabellen i en EXPRESS-modell är begränsad till 7-bitars ASCII. Data som beskrivs av en EXPRESS-modell ("utfallsrummet") kan självklart innehålla godtyckliga tecken.

```
ENTITY line;
     start_point: point_2D;
     end_point: point_2D;
     OPTIONAL line_width: REAL;
     OPTIONAL line_colour: colour;
     OPTIONAL line_font: font;
WHERE
     line_width >= 0;
END_ENTITY;

ENTITY point_2D;
     x: REAL;
     y: REAL;
END_ENTITY;

TYPE colour = ENUMERATION OF (red, blue, green, black);
END_TYPE;

TYPE font = ENUMERATION OF (solid, dash, dot);
END_TYPE;
```

Figur 3 Exempel på enkel EXPRESS-modell

EXPRESS-modeller skrivs med vilken editor som helst. Som ren ASCII är EXPRESS-modeller mycket lätta att flytta mellan olika system.

EXPRESS-G

I ett appendix till Part 11 definieras också EXPRESS-G, en grafisk form av EXPRESS. Syftet med EXPRESS-G är att göra det möjligt att visa modeller på ett mera visuellt och lättöverskådligt sätt.

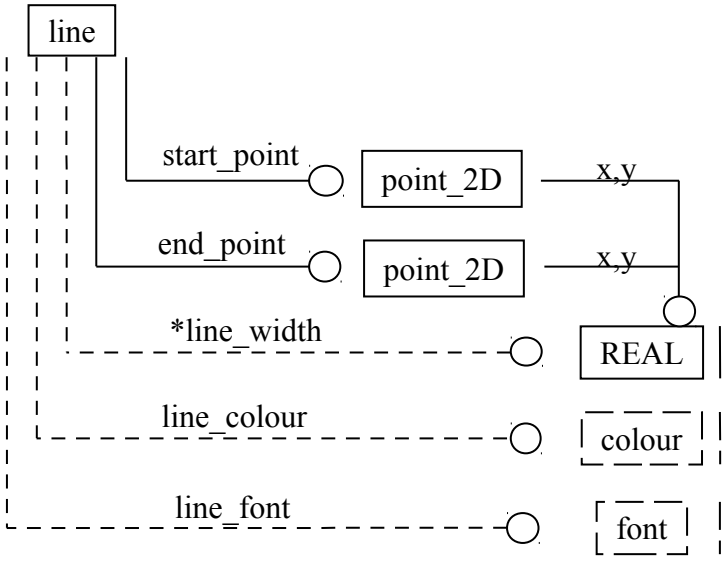
EXPRESS-G kan inte visa all information i EXPRESS-modellen, bland annat kan inte RULE-satser och WHERE-satser visas (ett *-tecken på attributnamnet används för att visa att attributet har ett WHERE-villkor, men själva villkoret kan alltså inte visas).

En bra EXPRESS-editor kan växla mellan att visa EXPRESS och EXPRESS-G. På så sätt får man både överskådlighet och fullständighet.

Det finns inget standardformat för EXPRESS-G-modeller. Det går alltså inte att flytta EXPRESS-G mellan olika miljöer. Man kan givetvis flytta modellerna som ren grafik men då har de tappat sin intelligens.

EXPRESS-G kräver i praktiken stora pappersytor för att det i verkligheten skall bli någon överskådlighet. A4 är fjuttigt på gränsen till meningslöst. A3 är bättre men bäst är givetvis ett rejält A0-lakan. A0-printrar är det inte alla som har tillgång till så det brukar bli en hel del klipp & klistra med A4-lappar. En användbar EXPRESS-G-editor måste alltså kunna skriva ut en stor modell uppdelad på flera A4-blad.

Har man inget krav på koppling mellan EXPRESS och EXPRESS-G-modell, kan man i princip använda vilket grafikprogram som helst för att rita EXPRESS-G-modeller.



Figur 4 Exempel på enkel modell skriven i EXPRESS-G

EXPRESS-X

EXPRESS-X är ett relativt senkommet tillägg till EXPRESS-familjen. Det är ingen viktig del av STEP men bör ändå nämnas för fullständighetens skull.

Syftet med EXPRESS-X är att kunna definiera kopplingar mellan olika EXPRESS-modeller. En STEP-intern användning är för att skriva mappningstabellen i ett AP i ett formellt och datortolkbart språk. En annan användning är för att definiera översättare för datamängder som följer olika EXPRESS-modeller.

Part 14 (CD) EXPRESS-X language reference manual.

EXPRESS-X är formellt sett en utvidgning av EXPRESS. I praktiken är det företaget STEP Tools Inc som ligger bakom EXPRESS-X.

Implementation med hjälp av filer

Filformatet är det klassiska sättet att överföra STEP-information. Man skickar filer mellan varandra, till exempel på diskett eller som en bifogad fil i ett e-post. Filformatet definieras i en separat delstandard:

Part 21 (IS) Clear text encoding of the exchange structure.

Formatet är definierat i form av en algoritm som utgår från någon utpekad EXPRESS-modell, till exempel EXPRESS-modellen i ett visst AP. Det exakta filformatet beror alltså på hur den utpekade EXPRESS-modellen ser ut.

Samma filformat kan alltså användas för samtliga AP. Detaljerna i filformatet kan också ändras utan att alla AP påverkas.

Filen består av två delar, HEADER-sektionen och DATA-sektionen. HEADER-sektionen innehåller lite information om filen, speciellt vilken EXPRESS-modell som används. HEADER-sektionen ser alltid likadan ut, den beror inte på EXPRESS-modellen. DATA-sektionen innehåller data enligt EXPRESS-modellen. En konsekvens blir att en mottagare som inte har tillgång till EXPRESS-modellen, inte kan tolka filen.

Inuti filen används löpnummer för att internt hålla reda på innehållets olika delar. Löpnumret skapas när filen genereras och har ingen betydelse utanför filen.

Olika operativsystem har olika konventioner för radslut och filslut. För att vara oberoende av operativsystem definieras filen som slut i och med sekvensen "`END-ISO-10303-21;`". En fil ur operativsystemets synvinkel kan alltså innehålla fler än en STEP-fil.

Filformatet använder teckenuppsättningen ISO 8859-1. Svenska tecken går alltså bra att överföra.

```
ISO-10303-21;

HEADER;
FILE_DESCRIPTION ('exempel på STEP-fil');
FILE_NAME ('linjebredd');
FILE_SCHEMA ('sample_draughting');
ENDSEC;

DATA;
#1 = line(0.0,0.0,1.0,1.0,0.065,black,dash);
ENDSEC;

END-ISO-10303-21;
```

Figur 5 Exempel på STEP-fil

Filformatet är inte särskilt effektivt ur lagringsteknisk synvinkel. Filerna tenderar att bli större än binärfiler och även relativt stora jämfört med andra liknande standardformat. Boven i dramat är entitetsnamnen som både är ymnigt förekommande i filen, samtidigt som de tenderar att vara rätt långa. För att undvika den värsta ineffektiviteten kan ett AP definiera förkortningar för entitetsnamnen. Filosofin bakom filformatet är som synes att prioritera läsbarhet och generalitet före kompakthet.

Redundansen i filen gör att den går bra att komprimera med något program typ WinZip eller Pkzip. När man skickar filer med e-post, har komprimeringen också den fördelen att man kan slinka förbi e-post-system som stuvar om bland å, ä och ö. Nackdelen med komprimering är att mottagaren måste packa upp filen.

Implementation med hjälp av ett programmeringsgränssnitt

SDAI (Standard Data Access Interface) är ett standardiserat programmeringsgränssnitt (Application Programming Interface, API) mot STEP-baserad information. Syftet med SDAI är bland annat att möjliggöra flyttbara applikationer.

SDAI ser i praktiken ut som ett klassbibliotek. En applikation anropar klasserna efter tycke och smak.

SDAI är i princip oberoende av vilket språk som används i applikationen men detaljerna i anropen ser lite olika ut. SDAI måste därför delas upp i en huvudstandard (Part 22) och språkspecifika delstandarder som definierar hur de språkspecifika bindningarna ser ut:

Part 22 (IS)	SDAI specification
Part 23 (IS)	C++ language binding to the SDAI
Part 24 (FDIS)	C language binding to the SDAI
Part 26 (CD)	IDL language binding to the SDAI
Part 27 (IS)	Java language binding to the SDAI
Part 29 (CD)	Light-weight Java language binding to the SDAI

Parameterlistan i ett anrop ges av EXPRESS-modellen. SDAI anropar alla enskilda data-element, till exempel alla attribut till en entitet, explicit. Inga faciliteter finns för att anropa hela entiteter eller grupper av entiteter. SDAI fungerar alltså på en rätt låg nivå. Fördelen är enkelheten. Nackdelarna är att programmeringen kan bli lite omständlig samt att prestanda kan bli dåligt om anropen tar relativt lång tid, till exempel om de sker över ett nätverk.

```
#include <stdio.h>
#include <sdai.h>

main()
{
blabla;

x = sdaiGetAttr(line_Lisa,line,start_point,x);

blablavidare;
}
```

Figur 6 Exempel på SDAI-anrop i C/C++

SDAI-implementationer är inte lika vanliga som fil-implementationer. Vanligaste användningsområdet är som API till renodlade STEP-databaser.

Implementation med hjälp av XML

Detta är en ny implementationsmetod som beskrivs i Part 28:

Part 28 (CD) XML representation for EXPRESS-driven data

Syftet är att göra STEP-information tillgänglig i Internet-anpassad form. XML är det självklara alternativet för detta. Grundtanken är alltså att använda XML som filformat, istället för Part 21. Sättet som man inom arbetsgruppen valt att göra detta, är att definiera översättningar från EXPRESS till XML DTD:er. Man har inte kunnat enas om en enda översättning, utan har inte mindre än fyra översättningar definierade, alla med olika för- och nackdelar. Part 28 är rätt hårt kritiserat och det är osäkert vad som kommer att hända framöver. Det vettigaste är nog att avvakta nästa försök, där man avser att definiera en standardiserad översättning från EXPRESS till XML Schema. XML Schema ligger närmare EXPRESS i uttrycksförmåga och det finns betydligt större förutsättningar för att man här skall lyckas åstadkomma något användbart.

Export av EXPRESS-modeller till UML

UML (Unified Modeling Language) har blivit en de facto standard för systemutveckling och systembeskrivning. Delar av UML, primärt klassdiagrammen, överlappar med den informationsmodellering som EXPRESS är specialiserat på. I vissa situationer kan det vara intressant att kunna lyfta över EXPRESS-modeller till ett UML-verktyg.

XMI (XML Metadate Interchange) är en kodning av UML-modeller i XML. Man kan alltså flytta UML-modeller mellan UML-verktyg via XMI. Part 25 definierar en översättning från EXPRESS till XMI.

Part 25 (AWI)	EXPRESS to OMG XMI binding

Stora delar av EXPRESS har ingen direkt motsvarighet i UML och översätts därför inte, till exempel regler. De mest grundläggande delarna av EXPRESS-koden, entitetsdefinitionerna, översätts och dyker alltså upp som klassdiagram i UML.

Konformitetstestning

Som användare vill man inte hålla på och testa STEP-överföringar. Helst skall det bara fungera utan problem.

Konformitetstestning (eng. Conformance Testing) bygger på idén att oberoende test-labb testar att STEP-mjukvaran följer standarden. Labbet kan sedan utfärda ett Certificate of Conformance. Som köpare kräver man att leverantören kan visa upp ett sådant cetrtifikat.

För att detta skall fungera krävs att det finns något entydigt sätt att testa mot STEP-standarden. Några delar av STEP (30-serien) sysslar med att definiera allmänt hur testningen går till.

Part 31 (IS)	General concepts
Part 32 (IS)	Requirements on testing laboratories and clients
Part 34 (IS)	Abstract test methods for Part 21 implementations
Part 35 (CD)	Abstract test methods for Part 22 implementations

Till varje AP finns det, eller borde finnas, en svit av testfall. Dessa definierar mer i detalj hur en mjukvara, som hävdar att den är en kompatibel implementation av ett AP, skall testas. Numreringen är densamma som AP-numreringen men nummerserien börjar med 301 (istället för 201).

ATS302 (WD)	Associative Draughting
ATS303 (WD)	Configuration Controlled Design
ATS304 (TS)	Mechanical Design Using Boundary Representations

ATS307 (TS) Sheet Metal Die Planning and Design

Som synes är tillgången på testsviter synnerligen mager. Orsaken är en kombination av resursbrist och brist på intresse. ATS:er är helt enkelt rätt trista, både som arbetsuppgifter och som finansieringsobjekt. Tills vidare har man därför ingen större nytta av Conformance Testing. Förlusten är kanske inte så stor, STEP verkar vara så välskrivet att tillförlitligheten blir bra ändå.

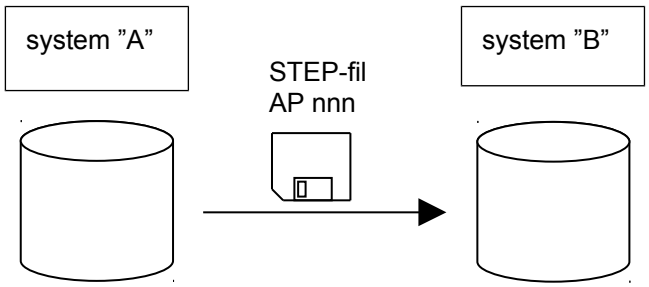
6 Användning

För att flytta filer

Man börjar med att det sändande systemet genererar en STEP-fil enligt något AP. Mjukvaran som gör detta brukar kallas pre-processor och finns i regel att köpa som tillbehör till systemet. I regel köper man en pre-processor för varje AP. Innan man genererar filen har man i regel rensat den från onödig information och har också allmänt kvalitetskontrollerat informationen.

På något sätt överförs filen till det mottagande systemet. Man kan till exempel använda band, diskett, Zip-diskett eller CD-R. Det kanske smidigaste sättet är att skicka filen som en bifogad fil med ett e-post-meddelande eller att lägga upp den för nerladdning på någon server (till exempel via http eller ftp).

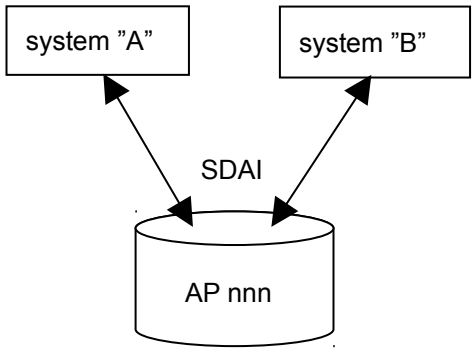
Det mottagande systemet läser in STEP-filen. Mjukvaran för detta kallas för post-processor. Även dessa finns som tillbehör till systemen och även här måste man kontrollera att den klarar det använda AP:et. Väl inläst i systemet kontrollerar mottagaren att allt ser bra ut.



Figur 7 Användning av STEP för att flytta filer mellan system

En nackdel med det här användningsättet är att man blandar ihop konvertering och transport. Om det blir fel, är det svårt att veta var felet uppstod, om det var i pre-processningen, i överföringen eller i post-processningen. För felsökning och kvalitetskontroll är det bra om man kan inspektera innehållet efter pre-processning och före post-processning. Bäst är om systemen själva klarar sådana kontroller, annars finns färdig sådan mjukvara att köpa från specialister på STEP-konverteringar.

Som en gemensam databas

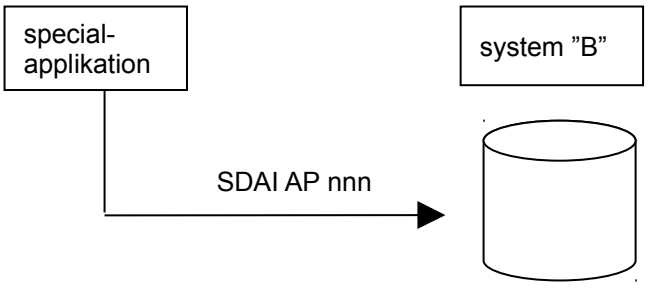


Figur 8 Användning av STEP som gemensam databas

I princip kan man bygga vad som helst ovanpå en ren STEP-databas. Man separerar alltså helt applikationen och dess data. Tanken kan ses som client/server på en högre abstraktionsnivå.

För flyttbara applikationer

En variant av den gemensamma databasen är när ett system tillhandahåller SDAI som ett API mot den egna (icke-standardiserade) databasen. Detta gör det möjligt att skriva special-applikationer utan att dessa blir låsta till något visst system.



Figur 9 Användning av STEP för flyttbara applikationer

När tillgängliga AP inte duger

Då är det i princip "kört". STEP:s existensberättigande är just sina AP. Att dra igång ett nytt AP inom ramen för ISO-arbetet är ett långsiktigt projekt. Att utveckla ett nytt AP behöver däremot inte vara särskilt dyrt, man gör ju det i samarbete med andra som har liknande behov. Det gäller mest att ha en långsiktighet i finansieringen och i intresset.

En annan framkomlig väg är att, utanför ISO-arbetet, definiera ett eget privat litet AP för sig själv och sina samarbetspartners. Man får göra allt själv men i många situationer skulle man fått göra det ändå, så varför inte utnyttja de metoder och verktyg som finns färdigutvecklade inom STEP-världen? Möjligheten finns alltid att det kan utvecklas till att längre fram bli ett reguljärt AP inom ISO, om man har nytta av det.

För applikationsutveckling

Inget hindrar att STEP-teknik, i synnerhet EXPRESS, används för klassisk applikationsutveckling. Man skriver egna EXPRESS-modeller, skapar en databas av modellerna och kopplar applikationerna till databasen med SDAI eller med filformatet.

Man har därmed skapat något utanför de AP som finns. Man är alltså inte längre kompatibel med STEP, man bara använder verktygen bakom STEP.

Fördelarna är att man kan återanvända befintliga EXPRESS-modeller samt att man kan använda EXPRESS och de styrkor som EXPRESS har.

Nackdelarna är att marknaden för EXPRESS-verktyg är liten jämfört med vanlig applikationsutveckling. Detta slår igenom i tillgång på verktyg och kunnigt folk, vilket i sin tur slår mot produktiviteten.

Vill man använda färdiga EXPRESS-modeller, kan man ofta göra det även i vanlig applikationsutveckling. Man översätter helt enkelt, manuellt om så krävs. Tiden eller kostnaden för översättningen är ofta i sammanhanget mycket marginell. Det avgörande är hela tiden den totala produktiviteten.

7 Alternativ eller komplement till STEP

Det finns alternativ och komplement till STEP. En kort genomgång av dessa kan vara intressant. Tabellen nedan ger en översiktlig bild av hur STEP förhåller sig till några andra företeelser.

	STEP	**XML**	**RDB**	**ODB**
presentation	-	CSS, XSL	-	-
icke-interaktiv åtkomst	STEP-filer	XML-filer	-	-
interaktiv åtkomst	SDAI	SAX, DOM, XQL	SQL	OQL
semantik/ innehåll	AP	XML-applikationer	-	-
språk för innehållsdef.	EXPRESS	DTD, XML Schema	SQL	ODL

XML

Syftet med XML (eXtensible Markup Language) är att skapa en efterföljare till mega-succén HTML. Man vill ha något som är mer generellt användbart och som inte lider av HTML:s begränsningar.

Tekniskt sett XML är en förenklad och Internet-anpassad version av SGML (Standard Generalized Markup Language, ISO 8879). SGML har under ett antal år används både för att beskriva och representera dokument.

XML har en syntax för att beskriva innehållet i ett dokument. Den syntaxen fungerar som en specifikation på ett filformat. En annan syntax används för att definiera regler för innehållet i ett dokument, dvs. något som kan liknas vid dess informationsmodell. En sådan (pseudo-) informatonsmodell kallas för en DTD (Document Type Definition). DTD-syntaxen motsvarar alltså ungefär EXPRESS. En DTD kan alltså ses som motsvarigheten till ett AP. Den mesta kända DTD:n är HTML. DTD:er duger för traditionella dokument men räcker inte för att definiera mer generellt användbara datatyper, på det sätt som till exempel EXPRESS gör. XML Schema är

under utveckling och är tänkt att komplettera eller helt ersätta DTD-syntaxen. XML Schema använder vanlig XML-syntax, istället för den speciella DTD-syntaxen. XML Schema har också betydligt bättre uttrycksfullhet, i princip likvärdig med EXPRESS.

Med XML vidgas definitionen på vad som är ett dokument. XML ser dokument helt enkelt som en klump information, med eller utan de egenskaper som traditionellt associeras med begreppet dokument (tryckbart, textorienterat etc.). XML har börjat tillämpas på områden som ligger inom STEP:s domäner. SVG (Scalable Vector Graphics) är en XML-baserad motsvarighet till AP201, med stöd från bl.a. Autodesk, Adobe och Microsoft. Användning av XML för 3D modeller har redan satt igång.

PDML (Product Data Markup Language) är ett mycket bra och intressant exempel på hur man kan använda STEP/EXPRESS i samverkan med XML. Bakom PDML ligger PDIT Inc., en amerikansk konsultfirma med inriktning mot STEP. Grundidéerna bakom PDML är att använda EXPRESS-modeller från STEP (PDM Schema) för att beskriva begreppen, i detta fallet PDM-relaterade begrepp, och att sedan använda XML för implementationen. Man utnyttjar exaktheten och generaliteten i EXPRESS-modellering samtidigt som tillgången på verktyg och spridningspotentialen hos XML används för implementationen. Tekniskt går det till så att man med en algoritm, implementerad i mjukvara, översätter från EXPRESS till XML DTD-syntax. Algoritmen och mjukvaran är inte särskilt märkvärdiga och behöver heller inte vara det i detta speciella fallet.

XML är alltså väldigt likt STEP och i princip likvärdigt i funktionalitet. Det intressanta med XML är att det har fått ett enormt genomslag på marknaden. Marknaden för XML-baserade verktyg är många tiopotenser gånger större än marknaden för STEP-verktyg. Detta gör att man måste betrakta XML som en de facto standard för den här typen av problem. Detta utsätter STEP för ett starkt konkurrenstryck. I den nya situationen är det upp till STEP att visa att man tillför något väsentligt, utöver vad XML klarar av. I dagsläget klarar man i stora delar av att göra detta, sett ur rent teknisk synvinkel. EXPRESS är ett mycket bra språk för informationsmodellering och det ligger enormt mycket arbete nerplöjt i alla modellerna. Hur det ser ut i framtiden är betydligt osäkrare. XML utvecklas intensivt och det är nog troligt att det kommer fram tekniskt intressanta alternativ till EXPRESS. Kommersiellt sett har XML för länge sedan vunnit matchen.

För många applikationer duger dagens XML alldeles utmärkt och man väljer därför självklart XML. Nästan alla nya initiativ använder nu XML och intresset för STEP har helt klart minskat. Part 28 kan ses som ett något yrvaket försök att hänga med i den snabba utvecklingen runt XML. Att kommittén bakom Part 28 inte har lyckats särskilt bra, är inte bra för STEP.

PDES Inc, i praktiken amerikanarnas STEP-organisation och en av stöttepelarna bakom STEP, har dragit igång något man kallar för STEPml. Man marknadsför det som "STEP för webben". Vad man gör är att man helt sonika plockar valda EXPRESS-modeller från olika AP och skriver om dom i XML-form (DTD eller XML Schema). Hur man gör den översättningen, är inte självklart. Det är det som man inte kunnat komma överens om inom Part 28. STEPml kör helt enkelt över ISO-processen och Part 28. Ur standardiserings-politisk synvinkel är detta inte helt oproblematiskt, för att uttrycka sig milt, men man har alltså inte tålamodet att vänta.

Egentligen finns det inget motsats- eller konkurrens-förhållande mellan STEP och XML. Det primär syftet med STEP är att tillhandhålla AP. Vilka verktyg och språk dessa använder, är av underordnad betydelse. Hade man påbörjat STEP idag, hade man självklart använt XML (och troligen också UML) internt, istället för Part 21 och

EXPRESS. STEPml må agera rätt brutalt men man gör nog i grund och botten helt rätt. STEPs framtid ligger i att använda XML på ett smart sätt.

UML

UML (Unified Modeling Language) används för systemutveckling. Man använder det för att bygga upp modeller som beskriver funktionaliteten och beteendet i ett system eller i en applikation. Användningsområdet är snarlikt EXPRESS, men inriktningen är på mjukvara, inte på fysiska detaljer.

UML har bara ambitionen att vara ett språk för systemmodellering, inte att vara ett format för att överföra olika systemmodeller mellan olika CASE-verktyg.

UML är ett stort och mångfacetterat språk. Det täcker avsevärt mycket större områden än EXPRESS. EXPRESS motsvarar ungefär klassdefinitionsdelen av UML. UML:s klassdefinitioner har inte alls den detaljerade uttrycksfullheten vad gäller datatyper som EXPRESS har. En klassdefinition kan preciseras med något som kallas OCL (Object Constraint Language) som ger en funktionalitet som närmar sig EXPRESS vad gäller typer och regler. Part 25 definierar en översättning från EXPRESS till UML klassdiagram via en XMI-fil. Part 25 innehåller ingen översättningar till OCL, inget heller sagt om hur man gör för att gå från UML till EXPRESS.

Relationsdatabaser (RDB)

SQL (Structured Query Language) är de facto-standarden för relationsdatabaser, vilken är den i särklass vanligaste organisationsformen för en modern databas. Ingen implementation skall göras utan att man åtminstone har tittat på RDB:er.

Passformen mellan STEP och RDB:er är tämligen dålig. STEP hör till den klassen av icke-reguljära komplexa datatyper som RDB:er har svårt att hantera. Största problemet brukar vara dåliga prestanda.

Det finns varianter av RDB:er där man bygger en objektdatabas ovanpå en RDB. Dessa kallas för hybriddatabaser eller "objectrelational databases". I vissa tillämpningar är detta en klart vettig idé. Man får den större semantiska uttrycksfullheten som databaser erbjuder, men kan fortsätta med att använda sina kära och välkända RDB:er. Det grundläggande problemet ur STEP-synvinkel, bristande prestanda, kvarstår dock.

Objektdatabaser (ODB)

Objektdatabaser passar naturligt för STEP:s komplexa datatyper. ODB:er används oftast på formen av en C++-databas, ofta tätt kopplad till en applikation och levererad ihop

med applikationen. Huvudsakliga marknadsnischen är snabba databaser för komplexa datatyper.

ODMG (Object Data Management Group) har definierat två ODL (Object Definition Language) och OQL (Object Query Language). Dessa standarder sågs i början som efterföljare till SQL men har inte alls fått samma genomslag.

Bibliografi

Länkar

SC4

STEP On-Line Information Server (SOLIS)	http://www.nist.gov/sc4/

Organisationer

PDES Inc.	http://pdesinc.aticorp.org
ProSTEP	http://www.prostep.de
SwedSTEP	http://www.psm.kth.se/swedstep/

Större projekt

POSC	http://www.posc.org
NIIIP	http://www.niiip.org
STEPml	http://www.stepml.org

Leverantörer av STEP-verktyg

STEP Tools Inc	http://www.steptools.com
EPM Technology A/S	http://www.epmtech.jotne.com
Conformics AB	http://www.conformics.com

Köp av STEP-standarder

ISO har copyright på alla dokument som har nått DIS eller IS-status. ISO tillhandahåller dem inte gratis utan säljer dem, faktiskt till ett väl tilltaget pris. ISO har heller inga rutiner för att hantera dem på elektronisk form utan är helt pappersbaserat i sin hantering och sitt tänkande. För vissa STEP-standarder, med omfång på uppemot 4000 sidor, blir situationen något besvärlig.

Man köper inte standarderna direkt av ISO, man köper dem av sin nationella standardiseringsmyndighet. Standarder på DIS-nivån kan man få via Charlotta Svartz på SIS, telefon 08-555 521 23 och e-post charlotta.svartz@sis.se. SIS tillhandahåller DIS-standarderna i pappersform till ett pris av 4 kr/sida eller på elektronisk form (PDF-fil på CD-R) till ett pris av 4500 kr per enskild delstandard. Priserna är ungefärliga. Standarder på FDIS eller ISO-nivån köper man genom SIS Förlag telefon 08-6103060. Standarderna kopieras upp på beställning och priserna ligger på några hundralapper upp till någon tusenlapp.

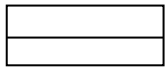
Böcker

"Information Modeling, The EXPRESS Way", Douglas Schenk & Peter Wilson, Oxford University Press 1994, ISBN 0-19-508714-3. Klassisk bok om EXPRESS, innehåller inget om STEP i övrigt.

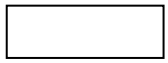
Appendix A: EXPRESS-G

EXPRESS-G används ofta för att visa innehållet i STEP:s informationsmodeller för folk som inte är kunniga i EXPRESS. Samtidigt är EXPRESS-G enkelt uppbyggt, varför det finns skäl för att ta med en mer komplett presentation, även i en mer introducerande bok som denna.

Grundläggande grafiska element:



schema



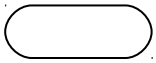
entity, type

(normal relation, "has a")

optional attribute, schema reference

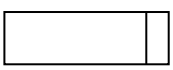
subtype/supertype

relation direction

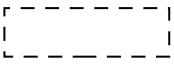


page reference (in multipage schemas)

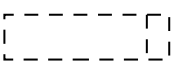
Data-typer:



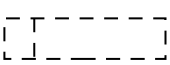
BOOLEAN, LOGICAL, BINARY, NUMBER, INTEGER, REAL, STRING



Defined Type

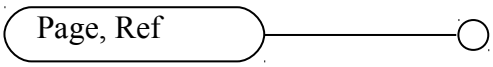


Enumeration Type



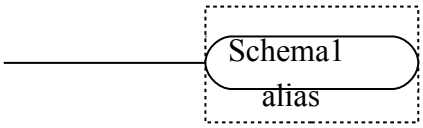
Select Type

Referenser till/från andra sidor eller schema:

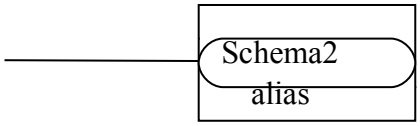


reference onto this page

reference to another page



referenced from another schema



used from another schema

Appendix B: Teknisk diskussion om STEP

Modulariseringen

Modulariseringen av STEP är relativt långt gången och man verkar ha gjort en vettig avvägning mellan flexibilitet och integration. STEP:s arkitektur har blivit stilbildande på detta område och några väsentliga ändringar är inte att vänta. Arkitekturen för de bakomliggande resursmodellerna är inte särskilt "ren" men det får man leva med.

Implementationsoberoendet

STEP är oberoende av vald implementationsteknik. STEP har inga synpunkter på vilket språk (C/C++, Visual Basic, Java etc.) eller vilken plattform (Unix, Windows etc.) som används för att bygga en STEP-implementation. Helt korrekt för en standard av den här typen.

Slutenheten gentemot omvärlden

STEP är uppbyggt helt och hållet med hjälp av en helt egenutvecklad teknik. EXPRESS, filformatet och SDAI är alla helt STEP-specifika konstruktioner. I långa loppet har detta varit en klok strategi. STEP har ett för långt tidsperspektiv för att det skall vara en vettig idé att anpassa sig till omvärldens för tillfället dominerande strömmningar. IGES var rätt kraftigt Fortran-influerat, vilket låg IGES i fatet när Fortran inte längre var standard. En tid fick STEP utstå en hel del kritik för att man inte använde den dåvarande de facto-standarden C++, vilket ju hade varit pinsamt att göra precis innan Java slog igenom. STEP har skapat sina egna verktyg och kan alltså segla vidare rätt ostört. Istället använder man modulariseringen till att skapa bryggor till dominerande branschstandarder som C, C++, Corba IDL och XML.

Statisk beskrivning

STEP bygger på existensen av föredefinierade och allmänt kända AP. Man måste känna till vilka AP som finns samt vad de innehåller och man får inte avvika från det. Skall man tolka en STEP-fil, måste man ha tillgång till EXPRESS-koden (i regel att AP) som filer hänvisar till. Detta innebär begränsningar. Dessa begränsningar är ofrånkomliga när man definierar en standard. På många sätt ligger standardens värde just i de begränsningar den sätter upp.

Skall man framföra en kritik mot STEP blir det mera på planet att STEP, i en strävan att vara alla till lags, tillåter för mycket. Olika AP är ibland mycket omfångsrika. En större tuffhet i urvalet av tillåtna begrepp hade nog kunnat ge både lägre komplexitet och högre tillförlitlighet. Den tuffheten är svår att upprätthålla i en konsensusbaserad process som ISO-processen.

Gör-det-själv för länkar

EXPRESS har inget färdigt begrepp för länkar. I STEP:s olika informationsmodeller finns heller inga allmänt vedertagna EXPRESS-modeller för att beskriva länkar. Det finns länkar av olika slag på några ställen men de är inte standardiserade. Slutresultatet blir att implementationer och verktyg har svårt att använda länkar.

Inget inbyggt tids- eller tillståndsbegrepp

Tanken bakom STEP är hela tiden att beskriva en produkt, underförstått en fysisk och tillverkad produkt. EXPRESS saknar färdiga begrepp för att beskriva dynamiska system, till exempel ett beteende eller en funktionalitet hos en produkt. Givetvis kan man bygga upp mycket av de begrepp som behövs i form av EXPRESS-modeller, men produktiviteten och effektiviteten kan bli sämre, jämfört med andra modelleringsspråk som är mer optimerade för det användningsområdet.