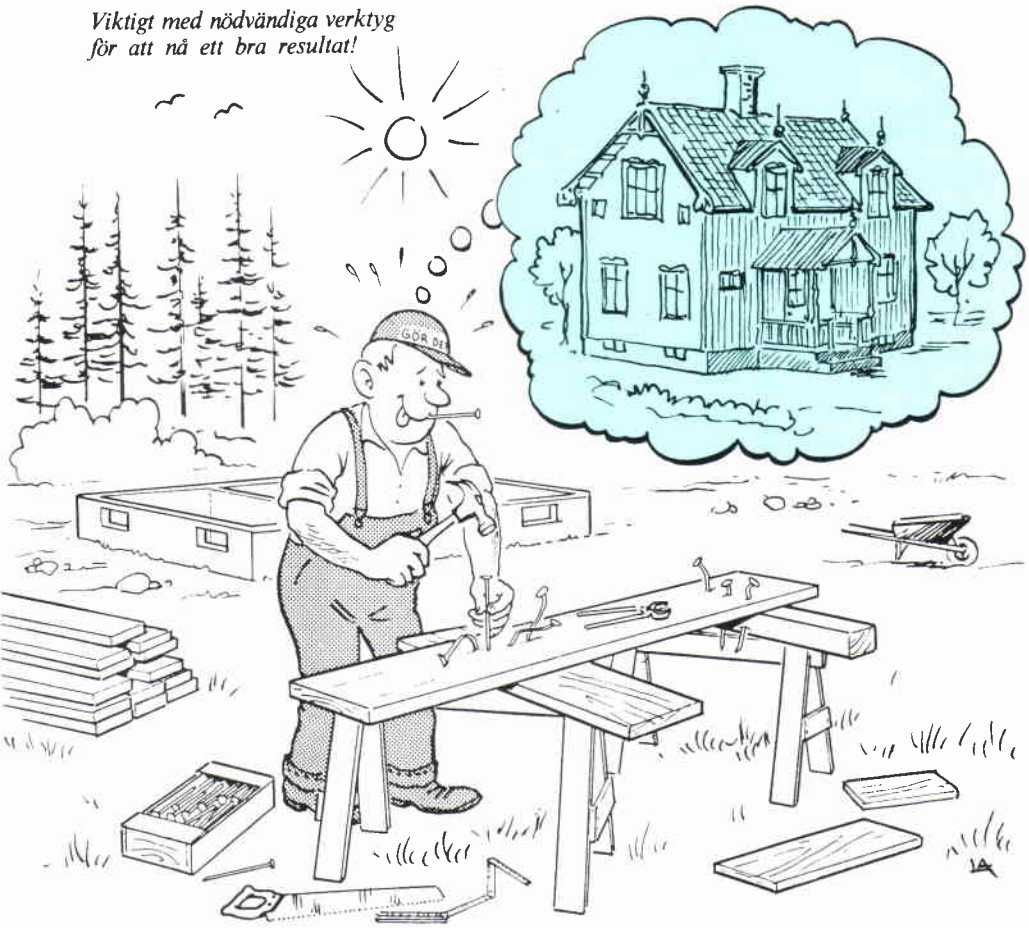
Viktigt med nödvändiga verktyg för att nå ett bra resultat!



Bara för att man lärt sig slå i spik med hammare kan man inte konstruera och bygga hus. Det räcker heller inte med att ha lärt sig koda i något programspråk för att kunna konstruera och tillverka programvara för datorer.

I förra TIFF behandlades kvalitetssäkring och kvalitetsstyrning av programvara. Här följer vi upp denna viktiga teknik med en presentation av Jan Jonsson, enheten för Autotestteknik vid FFV Underhåll. En metodhandbok för konstruktion av programvara är ett nödvändigt hjälpmedel för utveckling av programvara och utbildning av programmerare.

Text: ***Jan Jonsson*** *FFV-U/CVA Teckning:* ***Lennart Askerlöf,*** *FFV-U/CVA*

Programmeringsmetoder –

□ Datorer i olika former ingår numera allt oftare i produkter. I datorfunktionen ingår både maskin- och programvara. För att på ett säkert sätt uppfylla ställda fordringar på produkten under dess livstid, ställs krav på goda konstruktioner och hög kvalitet för såväl maskin- som programvara.

För maskinvara (mekanisk/elektronisk) finns i många fall etablerade metoder för konstruktion, komponentstandard m.m.

Ny teknikgren

Programvarukonstruktion och programvarutillverkning är dock en tämligen ny teknikgren. Behovet av metoder har på senare år blivit klart för de flesta i branschen. Internationellt pågår en intensiv debatt och utveckling av olika metoder för programvarukonstruktion.

De problem man vill undvika är numera nästan "klassiska" när det gäller framställning och leverans av programvara:

- Överskridna tidsramar
- Överskridna kostnadsramar
- Kvarstående fel i program efter leverans
- Felsöknings- och rättningsovänliga program
- Bristfällig dokumentation
- Kundens önskemål och krav ej helt uppfyllda.
- Kunden beroende av leverantören under lång tid.

Tidig erfarenhet

Redan i början av 1960-talet kom FFV Underhåll i kontakt med datorutrustade produkter, bl a automatiska testutrustningar (s k autotestare). För framtagning av programvara för dessa utvecklades riktlinjer i handboksform för konstruktion, framställning och dokumentering av testprogram, de s k PVA-instruktionerna. (PVA = Programvaruarbete.) Dessa instruktioner används fortfarande.

Nya behov

Tillverkning och underhåll av mera generell programvara (operativsystem, systemprogram, databehandlingsprogram etc) ökar dock alltmer. Som en följd av detta har också behovet av metoder för mera generell programvarukonstruktion ökat.

Detta har lett till att man inom FFV Underhåll tar fram "METODHANDBOK FÖR SYSTEM- OCH PROGRAMUTVECKLING".

Metodhandbok

Metodhandboken innehåller både allmänna riktlinjer och mera konkreta metoder och hjälpmedel för programvarukonstruktion. Följande ämnesområden och metoder finns nu eller kommer att införas inom en snar framtid:

SYSTEMUTVECKLING
- Mall för systemutveckling
- Systemspecifikation enligt mall

PROGRAMUTVECKLING
- Mall för programutveckling.
- Programspecifikation enligt mall
- Strukturerad programmering
- Tillämpad kodningsteknik

DOKUMENTATIONSSYSTEM
- Hantering av dokumentationssystem

Riktlinjer för kunddokumentation

SPECIELLA TEKNIKER
Strukturdiagram
Människa – maskin – kommunikation

Dessutom ingår testmetoder, hjälpprogram för programframställning, beskrivningar av programbibliotek (med generella subprogram som utvecklats internt) och hantering av datamedia (skivor, magnetband).

På sikt kommer det även att ingå metoder för projektledning och kontroll, riktlinjer för s k "walk-throughs" (genomgång av programvara inom en arbetsgrupp), kvalitetskontroll för programvara, sorteringsmetoder etc.

Bokens syfte

Tanken med metodhandboken är inte att skapa en massa restriktioner och regler som slaviskt måste följas till punkt och pricka. Den ska i stället ge ett *stöd* i arbetet och underlätta detta! Man ska inte behöva "uppfinna hjulet" varje gång man behöver det. Därför ska metoderna och hjälpmedlen ha två viktiga egenskaper, de ska vara:

Praktiska – visa sig fungera väl i verkligheten.

Färdiga – direkt användbara i det dagliga arbetet.

metoder i metodhandboken och sambandet mellan dessa.

Strukturdiagram

Ett strukturdiagram är en bild av hur något är uppbyggt eller uppdelat logiskt sett. Man kan rita systemstrukturer, programstrukturer, datastrukturer, dokumentationsstrukturer etc. Metoden är generell och används genomgående i metodhandboken. Den används för att *beskriva mallar* för system- och programutveckling eller för att *utveckla system och program.*

Den används även för att planera projekt eller undervisning. Figuren här är ett exempel på strukturdiagram. Men dessa kan på ett överskådligt sätt även beskriva mera komplexa och generella samband.

Mall för programutveckling.

Arbetet sker stegvis och delas upp i fyra faser. Varje fas avslutas och *dokumenteras* innan nästa påbörjas.

Programspecifikation enligt mall

En programspecifikation bestämmer vad ett program ska göra, men inte hur detta ska åstadkommas. Programmet ses som en "svart låda". En uppsättning *standardrubriker* underlättar utarbetandet av specifikationen och tjänar dessutom som "checklista" så att inget blir bortglömt.

Strukturerad programmering

Man utarbetar en programstruktur som visar vilka funktioner, delfunktioner och operationer programmet ska bestå av för att få specificerade egenskaper enligt fas 1. (Programstruktur ersätter "forna dagars" flödesplan).

Tillämpad kodningsteknik

Metoden beskriver hur en programlista ska utformas med kommentarer och inledande beskrivning samt redigeras för att bli överskådlig och lättbegriplig. Anger även hur variabelnamn, lägesnamn och filnamn bör väljas samt hur olika specialfunktioner kan kodas i olika programmeringsspråk.

Testmetoder

Anger tillgängliga tekniker och hjälpmedel för programutprovning.

Resultat

Vad vill man då åstadkomma med alla metoder och hjälpmedel? Primärt vill man uppnå följande effekter:

- Färre fel under arbetets gång och därmed mindre behov av att behöva gå tillbaka och göra om.
- Kortare utprovningstider, kortare projekttider.
- Högre produktivitet per person.
- Få eller inga kvarstående fel
- Bättre strukturer, bättre dokumentation och därmed kontrollerbara och modifieringsvänliga produkter.
- Minskat behov av system- och programunderhåll.
- Bättre möjlighet att utbilda nyanställda.
- Större möjlighet till projektstyrning.
- Möjlighet till kvalitetskontroll under arbetets gång, både från projektledningens och kundens sida.
- Ökad förmåga att tillgodose kundens krav och önskemål.

Detta i sin tur bör ju klara av de "klassiska problemen" enligt tidigare! Men inte bara det. En användbar och levande metodhandbok för system- och programutveckling kommer över huvud taget att ge det egna företaget en större styrka och kapacitet inom programvarukonstruktionens område, såväl kvalitativt som kvantitativt! En styrka och kapacitet som även kommer företagets kunder till del.

Här beskriven metodhandbok är ett internt dokument inom FFV för utveckling av programvara för bland annat försvarets räkning samt utbildning av egen personal. Boken kan delvis användas om det blir aktuellt även vid extern utbildning. ■

nödvändigt verktyg

Dessutom ska metoderna och hjälpmedlen i stor utsträckning vara *gemensamma.*

Metodutveckling

Detta gör att man inte utan vidare kan ta redan existerande "yttre" metoder och sätta in dem i metodhandboken.

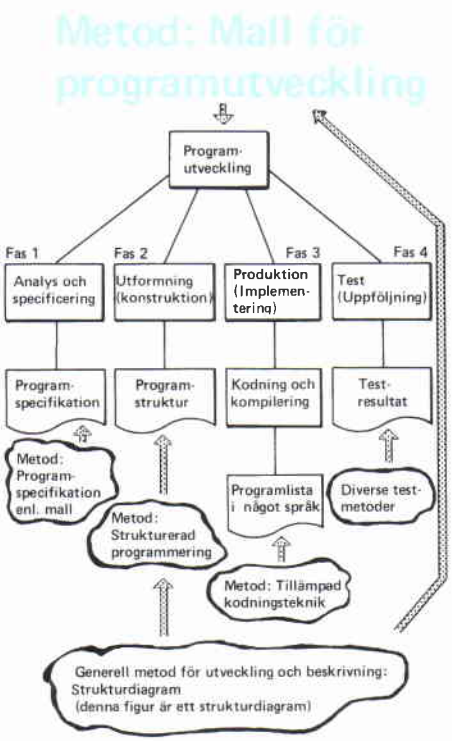
För det första måste man *välja ut* (och samordna) de metoder som passar bäst (och som inte är motstridiga). För det andra måste metoderna väldigt ofta justeras, anpassas eller vidareutvecklas.

I detta arbete tar man då hänsyn till den egna arbetssituationen och "bakar in" egna erfarenheter och önskemål. Ibland kan helt nya metoder och hjälpmedel behöva utvecklas internt.

För det tredje måste metoderna *spridas* bland berörda medarbetare om det ska vara någon mening med dem, helst genom en pedagogiskt genomförd utbildning. För det fjärde måste metodernas användbarhet följas upp, vid behov måste de omarbetas.

Metodexempel

Låt oss nu helt kort titta på några



Strukturerad programmering

En teknik som används för att höja programkvaliteten är STRUKTURERAD PROGRAMMERING. Rätt använd ger denna bättre konstruktion, läsbarhet, underhållsmässighet, standardiserad konstruktionsprincip och tillförlitligare program.

Text: Lars Finnström
FFV-U/CVA

Fig 1. Sekvens

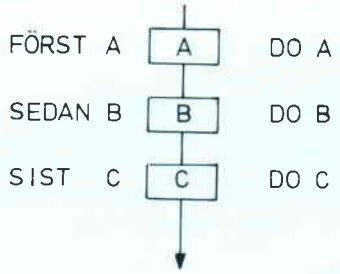


Fig 2. Selektion med två alternativ

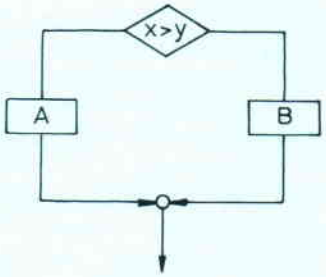


Fig 3. Iteration

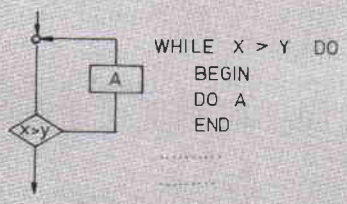


Fig 4. Selektion med fyra alternativ

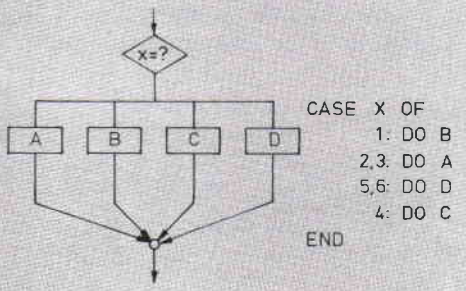
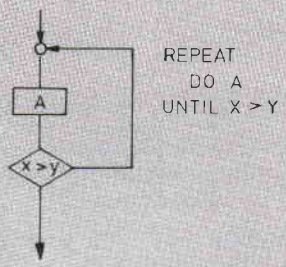
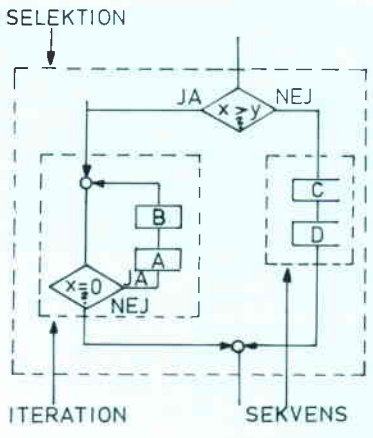


Fig 5. Upprepning






```
IF X > Y
THEN
    BEGIN
        WHILE X = 0 DO
            BEGIN
                DO A
                DO B
            END
    END
ELSE
    BEGIN
        DO C
        DO D
    END
```

Fig 6. Kombination av tre grundläggande komponenter

Verklighetens struktur

Hur verklighetens struktur ser ut kan belysas med ett enkelt exempel – en

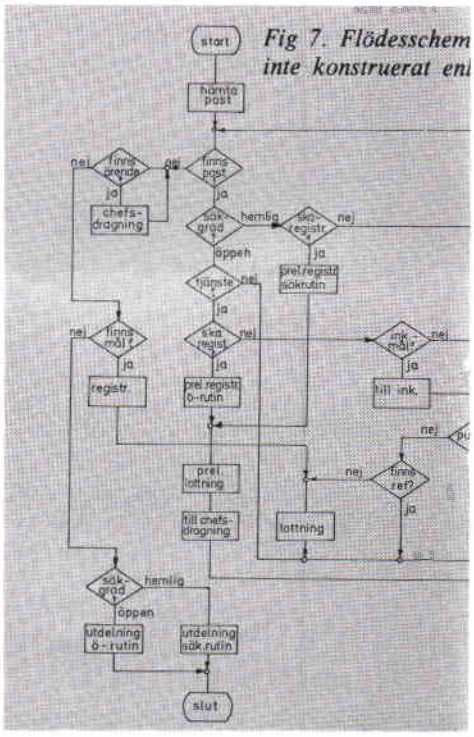


Fig 7. Flödesschem inte konstruerat en

postrutin inom en organisation eller företag. Först: Hämta post på postkontoret.

Sedan: Sortera
Sedan: Preliminär registrering och lottning.
Sedan: Chefsdragning.
Sedan: Slutlig registrering och lottning.
Sist: Utdelning

Postrutinen (förenklad) används som exempel dels för att den är lätt att förstå och dels för att visa att STRUKTURER finns överallt och inte bara i datorprogram.

Byggstenar

Studerar man "postrutinen" finner man att händelserna kommer efter varandra, först, sedan, sedan . . ., sist. Händelserna kommer i SEKVENS. Överfört i diagramform ser en SEKVENS ut som fig 1.

I fig 1 ser vi hur man först "Utför A = Do A", sedan "Utför B = Do B" och sist "Utför C = Do C". (Engelska brukar användas i dessa sammanhang.) A, B och C kan innebära vad som helst.

SEKVENS är den första av de byggstenar eller KOMPONENTER som får användas och står också för den kanske viktigaste principen nämligen att man gör saker i tur och ordning d v s SEKVENS.

Den andra byggstenen heter SELEKTION. Se fig 2. Här utförs ett val nämligen om A eller B ska utföras. Vilken av A eller B som utförs beror på värdena hos x och y. Om x är större än y utförs A. Är x lika med eller mindre än y utförs B. Ur fig 2 framgår också att det endast får finnas en ingång och en utgång från en byggsten eller KOMPONENT som den också kallas.

Den tredje KOMPONENTEN är ITERATION, se fig 3. Grundprincipen är att A ska kunna genomlöpas noll gånger. Det betyder att testen på villkoret måste komma först. ITERATIONEN ska inte förväxlas med upprepning där ofta testen görs sedan "A" genomlöpts en gång.

De tre beskrivna komponenterna (grundstenarna) kan åstadkomma allt som kan krävas i ett program. Används de och noteras på rätt sätt i programlistorna finns goda möjligheter att flera än programmeraren kan läsa och förstå programmet. Detta har stor betydelse om ändringar behöver göras.

Ytterligare två komponenter brukar användas vid SP nämligen SELEKTION mellan flera alternativ enligt fig 4 och UPPREPNING enligt fig 5.

Fig 6 visar en kombination av de tre grundläggande komponenterna. Den överordnade komponenten är en SELEKTION som testar x med avseende på y, om ITERATIONEN eller SEKVENSEN ska utföras. I ITERATIONEN modifieras värdet på x i A och/eller B tills värdet på x inte längre är noll, innan ITERATIONEN är slutförd.

Märk att utgång från den övergripande SELEKTIONEN måste ske i en punkt. Det är inte tillåtet att "hoppa" från ITERATIONEN till ett ställe i programmet och från SEKVENSEN till ett annat ställe. Dock kan ITERATIONEN följas av ytterligare KOMPONENTER t ex en SEKVENS, innan utgång från SELEKTIONEN görs.

Texten till höger i fig 6 visar hur fig uttrycks i högnivåspråket PASCAL. Skrivsättet kan också användas som PSEUDOKOD (eng Logikkod) för att förklara logiken innan kodning i aktuellt språk börjar. SP är alltså SPRÅK-OBEROENDE. Vissa språk är dock lättare att använda än andra vid SP, och andra är konstruerade för att ge SP, t ex Pascal.

Strukturerat – ostrukturerat

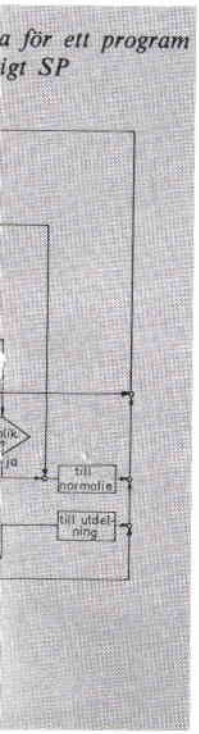
SP brukar ibland kallas GOTO-lös programmering. Detta är bara delvis riktigt. Av vad som framgått tidigare behövs inga GOTO, men i vissa språk t ex Fortran måste de användas, men endast i syftet att uppnå SP.

Ett flödesschema för ett program som inte är konstruerat enligt SP visas i fig 7. Beskrivning av funktionen kan inte göras här men håll med om att den inte är lätt att reda ut.

Samma program konstruerat enligt SP får ett strukturdiagram, inte flödesschema, enligt fig 8. Här är programlogiken anpassad till verkligheten och klart angiven. Ska ändringar införas är det lätt att se var rätta "platsen" är och vilka beroenden som finns till andra funktioner i programmet. Fig 8 är ritad enligt en princip som utvecklats av M Jackson, JSP, och innehåller endast de grundläggande komponenterna.

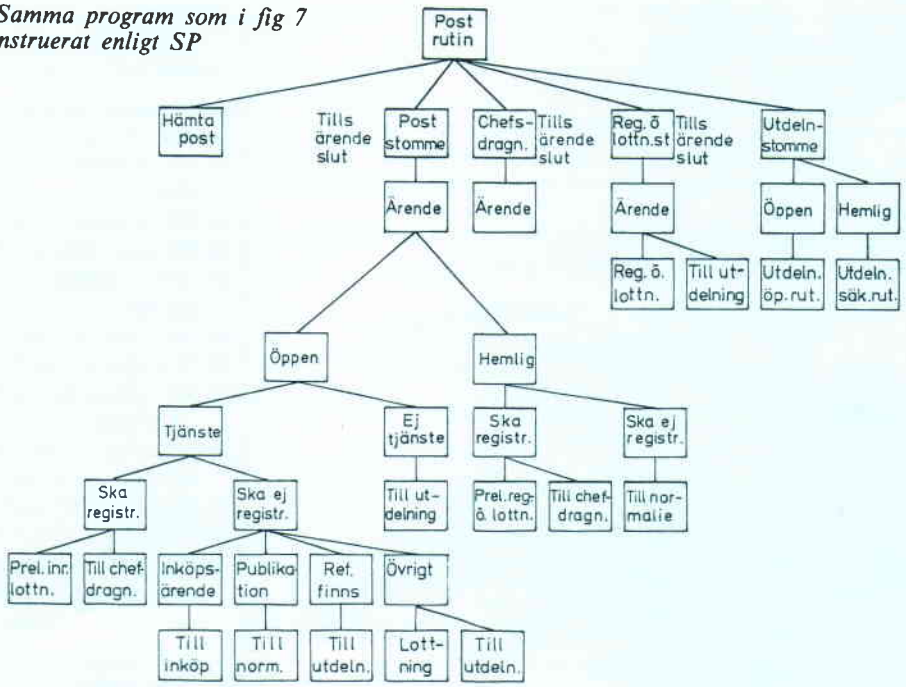
Stegförskjutning

I fig 8 framgår att diagrammet är uppdelat i nivåer, totalt 9 st. För att lättare kunna läsa programkoden mot strukturdiagrammet görs stegförskjutning åt höger i programkodlistan, så



a för ett program igt SP

Fig 8. Samma program som i fig 7 men konstruerat enligt SP



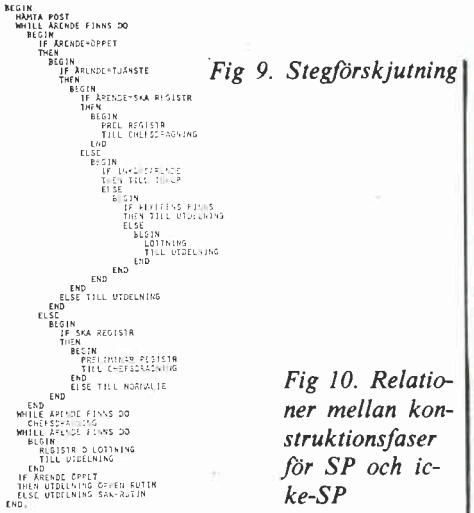


Fig 9. Stegförskjutning

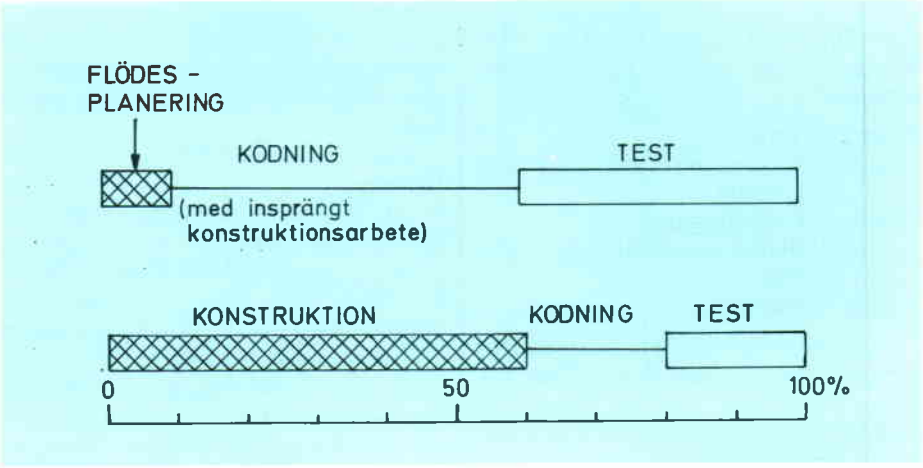


Fig 10. Relationer mellan konstruktionsfaser för SP och icke-SP

att varje steg svarar mot en nivå i diagrammet.

I fig 9 framgår stegförskjutningen tydligt. Programkoden svarar mot strukturdiagrammet i fig 8.

Datorprogram är dyra

Att konstruera datorprogram är en löneintensiv verksamhet och följaktligen mycket dyr. Att effektivisera konstruktionsarbetet betalar sig. Genom att använda SP kan kostnaderna hållas lägre än annars, och dessutom blir programmen tillförlitligare. Detta beror på att konstruktionen måste göras innan kodningen tar vid. Man måste tänka efter före. Hur mycket lägre kostnaderna blir för SP beror bl a på projektens storlek. Även om kostnaden för själva konstruktionen inte blir lägre sjunker livstidskostnaden p g a färre ändringar.

Det är vanligtvis testningen som ökar utanför godtagbara ramar för icke-SP, eftersom felen upptäcks här. I värsta fall måste programmet konstrueras om.

Relationer mellan konstruktionsfaserna för SP och icke-SP framgår av fig 10.

När SP inte hjälper

Det finns dock tillfällen där program konstruerade enligt SP inte hjälper.

Det första som samtidigt är det vanligaste vid all programkonstruktion är att programspecifikationen innehåller felaktigheter.

Det andra är att programmet har brister i strukturen. Vare sig det beror på programmerarens utbildning eller motvilja mot SP blir inte ett "halvstrukturerat" program ett STRUKTURERAT PROGRAM. ■

Beslut om livstidsförlängning av Hkp 4A och 4B

Text: ***Gert Almqvist FMV-FF/FL***
Foto: ***Barbro Bergström FFV-U/CVM***

Chefen för Marinen och chefen för Flygvapnet har beslutat förlänga livstiden för Hkp 4A och B till 1989/90 vilket innebär att helikoptrarna får nästan 30 års livstid.

□ För att kunna förlänga helikoptrarnas livstid med så många år erfordras en omfattande genomgång av struktur och system – främst elsystem. En viss modifiering av utrustningar måste även äga rum för att helikoptern ska kunna fullfölja förelagda uppgifter på ett bättre och säkrare sätt. För att få ett beslutsunderlag för serieinstallation har en provmodifiering av en Hkp 4A utförts vid FFV-U/L. Installationen som omfattat en mjukvarudel (framtagning av konstruktionsunderlag, typarbete etc) och en hårdvarudel (installationer, reparationer, H-ts etc) har pågått sedan februari 1981 och avslutades i planerad tid vecka 8 1982. Helikoptern har då varit på FFV underhåll i Linköping under 26 veckor.

Provinstallationen har förutom en noggrann detaljgenomgång av strukturen med utbyte av defekta delar och omnitning av uppglappade nitförband m m samt genomgång av samtliga system med utbyte av åldrande delar även omfattat byte av elledningar och skarvdon.

För övrigt har en modifiering av helikoptern skett genom införandet av:

- Automatisk länspumpsystem
- Ny typ av dränerventil
- Ny färdmekanikerstol och ytbärgarestol
- Modifierad räddningsvinsch
- Balanseringssystem för rotorerna
- Ny tätning av golvpaneler och luckor
- Ny typ av antikollisionsljus
- Ny typ av hydraulfilter
- Ny typ av instrumentomformare och likriktare
- Ytterligare en tändbox och tändkabel till varje motor
- Radarhöjdmätare nr 2
- Utbyte av Fr 12 till Fr 28
- Ny turbintemperaturindikator mer analog och med digitalavläsning samt med minne.

Utöver dessa punkter kommer ett helt nytt motorregleringssystem att införas.

FMV projektledare Gert Almqvist och FFV underhåll projektledare Rune Weidar flankerande besättningen från Fc och F 21 efter avslutad kontrollflygning.