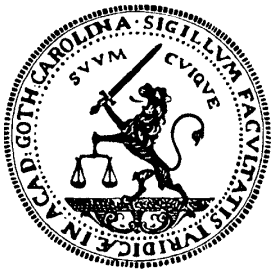

JURIDISKA FAKULTETEN
vid Lunds universitet

Erik Nordström

Plagiatvärderingen av datorprogram

Examensarbete
20 poäng

Handledare
Tf. Professor Hans Henrik Lidgard

Ämnesområde
Immaterialrätt

VT 1999

Innehåll

Sammanfattning

Datorprogrammens introduktion på den juridiska arenan föregicks av en debatt om den bästa skyddsformen, en debatt som inte är avslutad idag. Sedan mitten av 1980- talet skyddas datorprogram upphovsrättsligt, som litterära verk, såvida de traditionella kriterierna är uppfyllda. Datorprogrammens funktionsbindning och starka prägel av kvalitet leder dock till att det är svårt att applicera grundläggande begrepp, såsom verkshöjd, på datorprogrammen. Kvalitetsprägeln grundar sig på att mottagaren av verket är en dator och inte en människa. Kvalitetsaspekterna hör således inte samman med det estetiska i verket utan med de kvalitativa egenskaper som en dator utgår från, såsom programmets minnessnåla struktur eller eleganta lösningsmetoder. Funktionsbindningen består i att hänsynstagande måste tas till de begränsningar programspråket sätter för att uttrycka den avsedda funktionen. Kvaliteten på ett verk är inom den traditionella upphovsrätten inte en faktor som påverkar skyddsbarheten av verket, vilket innebär en konflikt med de inneboende kvalitetsaspekterna i ett datorprogram. Det skyddsvärda i programmet återspeglar sig i programmets kvalitet och borde därför vara en faktor som bestämmer skyddsomfånget för datorprogrammet. Preambeln i EG- direktivet för skydd för datorprogram stadgar att kvalitetsaspekten inte skall ligga till grund för originalitetsbedömningen men detta torde endast vara relevant vid skyddsbarheten och ej skyddsomfånget.

I intrångsprocesser som rör traditionella litterära verk kan en direkt jämförelse göras mellan texterna för att avgöra om olovlig exemplarframställning har skett. Rörande datorprogram kan denna jämförelse inte göras på samma sätt, detta för att programmen kan vara skrivna i olika programspråk. Skillnaderna mellan programspråken är ibland så stor att den inte kan jämställas med en översättning enligt UrL 4§ 2 st. Frågan har därför uppkommit i amerikansk praxis huruvida de icke- litterära elementen i ett program, såsom strukturen och organisationen av modulerna, (SSO), kan åtnjuta skydd. För att en jämförelse skall kunna göras mellan programmen har den sk Trestegsprocessen utvecklats som ett hjälpmedel. Processens första steg abstraherar funktionen i varje del av programmet. Därefter filtreras element som är dikterade av effektivitet eller externa faktorer, varpå delarna jämförs med de påstådda plagierade delarna. Metoden är ofta tillämpad i USA men problemet är att det finns risk att strukturer med kvalitativ hög nivå, dvs effektiva strukturer, konstant drabbas. Varken EG- direktivet eller de svenska förarbetena nämner något om skyddet för SSO. Det skyddsvärda i ett program återfinns till stor del i hur programmet är strukturerat. Ett hinder för skydd för SSO borde därför inte föreligga.

Det visuella gränssnittet i ett program utgörs av det som användaren ser på bildskärmen. Datorprogram skyddas som ett verk vilket innebär att alla dess

framträdelseformer är skyddade. Originalitetsbedömningen får därför även göras utifrån det som programkoden genererar. Utformningar som är knutna till dess funktion att styra programmet åtnjuter inte skydd, med undantaget att sk pionjärutformningar kan åtnjuta skydd.

De metoder som utvecklas för plagiatvärderingen av program är inte beständiga. Den tekniska utvecklingens frammarsch inom programmeringen innebär att nya metoder måste utvecklas för att kunna appliceras på programmen. Nästa generations språk inom programmeringen, med de struktureringstekniker de innebär, kommer att leda till att nya metoder utvecklas. Debatten om plagiatvärderingen för datorprogram är således inte avslutad.

Förkortningar

EG-direktivet	Rådsdirektiv 91/250/EEG av den 12.5.1991 om skydd för datorprogram
F. Supp	Federal Supplement
HD	Högsta Domstolen
HovR	Hovrätten
IIC	International Review of Industrial Property and Copyright
MFL	Marknadsföringslag (1995:450)
NIR	Nordiskt Immateriellt Rättsskydd
NJA	Nytt Juridiskt Arkiv
Rt	Norsk Retstidende
S. D. N. Y	Southern District of New York
SOU	Statens Offentliga Utredningar
UfR	Ugeskrift for Retsvaesen
UrL	Lag (1960:729) om upphovsrätt till litterära och konstnärliga verk
U. S.	United States
U. S. C.	United States Code
U. S. P. Q.	United States Patent Quarterly
WIPO	World Intellectual Property Organisation

1. Inledning.

Sedan 1980- talet är datorprogram definierat som en verkskategori i UrL, lagen om upphovsrätt. Lagändringen föregicks av en intensiv diskussion huruvida upphovsrättsligt skydd verkligen var den bästa skyddsformen för datorprogram och diskussionen fortgår än idag. Syftet med lagen är att skydda, och därmed indirekt stimulera, andligt skapande. Då den tekniska utvecklingen är ytterst svår att förutsäga, har lagen gjorts teknikoberoende, dvs lagen kännetecknas av generella stadgande, avsedda att även kunna appliceras på nya verksformer som återspeglar andligt skapande. Detta är på både gott och ont. Med generella stadgande skyddas, utan närmare definition, praktiskt taget allt som är resultatet av kreativt skapande, samtidigt som osäkerhet uppkommer när nya verkskategorier infinner sig. Förarbeten och praxis, både nationell och gemenskapsrättslig sådan, har därför en betydande roll inom upphovsrätten.
Datorprogrammen, som är klassificerat som ett litterärt verk, skiljer sig i många avseenden från de traditionella litterära verken. Dess uppgift är att lösa specifika uppgifter för användaren och har därför en tydlig funktionell prägel. Datorprogrammens inträde på den juridiska arenan har satt de grundläggande upphovsrättsliga begreppen på prov och trots att verkskategorin inte är särskilt ny, har än idag inga konkreta ramar utkristalliserats. Detta beror främst på att relevant praxis i stor utsträckning saknas i Sverige men även på att EG- domstolen inte har fått anledning att pröva innebörden av EG- direktivet för datorprogram. I stället har blickarna vänts mot USA vars domstolar, ända sedan datorprogrammens initialskede, i åtskilliga fall har prövat copyrightskyddet för datorprogram. Den fortskridande utvecklingen inom programmeringstekniken leder dock till att metoder anpassade för ett specifikt fall kullkastas och ersätts med nya i linje med den rådande verkligheten.

1.1 Syfte.

Uppsatsen behandlar främst två frågeställningar, dels hur jämför man två datorprogram i en intrångsprocess, dels om även de icke- litterära elementen i ett datorprogram kan skyddas. I diskussionen kring den första frågeställningen kommer jag att redogöra för de element i ett datorprogram som kan vara av vikt för jämförelsen, såsom den rena kodtexten, hjälptexter, genererade effekter, strukturering av programmet etc.

Diskussionen kring plagiatvärderingen leder in på frågeställningen huruvida strukturerna i ett datorprogram kan åtnjuta skydd. Den traditionella upphovsrätten för litterära verk är fokuserad på de själva litterära elementen i verket. Ett datorprogram är mer komplext i det avseendet att de litterära elementen genererar en avsedd effekt. De inbördes relationerna mellan byggstenarna i programmet styr i samma omfattning de genererade effekterna som den själva litterära programkoden. Jag kommer att föra en

diskussion, med bakgrund av de knapphändiga förarbeten och EG- direktivet samt genom att dra paralleller med amerikansk rättspraxis, huruvida de icke-litterära elementen i datorprogrammen kan åtnjuta upphovsrättsligt skydd. I detta avsnitt kommer jag även att redogöra för den metod som används främst i USA för att jämföra datorprogram som inte är skrivna i samma programspråk, den sk Trestegsprocessen.

1.2 Disposition

Jag kommer att inleda med en redogörelse för diskussionen som föregick införandet av datorprogram som en verkskategori, detta för att belysa de skillnader som finns gentemot de traditionella litterära verken. Detta avhandlas i kapitel 4. I kapitel 6 utreder jag verkshöjdsbegreppet för datorprogram med bakgrund av den nya tolkningskällan i form av EG-direktivet. I detta kapitel kommer jag att redogöra för de svårigheter som finns med att applicera traditionella upphovsrättsliga begrepp på datorprogram och undersöka om inte nya aspekter måste vägas in i bedömningen om verket åtnjuter skydd. I kapitel 7, som rör plagiatvärderingen, undersöker jag om internationell praxis, främst amerikansk, har någon relevans för svenska och europeiska förhållanden. I kapitel 7 för jag även en diskussion huruvida de strukturella elementen i datorprogram kan åtnjuta skydd. Uppsatsen inleds med en kort teknisk inledning, knuten till grundläggande begrepp inom programmering, för att underlätta förståelsen av uppsatsen. Trots att kontinuerliga sammanfattningar kommer att ske efter varje avsnitt avlutas uppsatsen med ett avsnitt med sammanfattande slutsatser.

1.3 Metod och materialval

Uppsatsen bygger på litteratur- , rättsfalls och artikelstudier, som utgör grunden till uppsatsens deskriptiva moment. I de avseenden jag ansett att det har behövts har jag i anslutning till de deskriptiva momenten analyserat intressanta frågeställningar och redogjort för mina egna slutsatser. De komparativa elementen i uppsatsen består av studier av amerikansk praxis. I detta sammanhang sker en analys av i vilken omfattning denna praxis har betydelse både för svenska och europeiska förhållanden.

Materialet har sitt huvudsakliga ursprung från nordiska källor, både i form av litteratur och artiklar samt nordisk och internationell rättspraxis. Med tanke på att LEXIS inte längre är tillgängligt på Lunds bibliotek, har jag även använt mig av Internet för artikelsökningar med de begränsningar som mediet innebär. Det skall sägas att den begränsade möjligheten att finna amerikansk litteratur och artiklar, i viss mån har tvingat mig att använda mig av andrahandskällor.

1.4 Avgränsningar

Uppsatsen ämnar endast att behandla de immaterialrättsliga aspekterna av datorprogram. Således behandlas inte det konkurrensrättliga perspektivet som kan läggas på ämnesvalet, vilket innebär att de amerikanska antitrust lagarna inte kommer att behandlas. Datorprogram kan i viss utsträckning numera åtnjuta patentskydd i USA men detta kommer heller inte att behandlas. Lagen om skydd för företagshemligheter kan ingripa på de fall där arbetstagaren uppträder illojalt mot arbetsgivaren. Lagen är ofta tänkbar i situationer som rör illojalt handskade av datorprogram i anställningsförhållande men detta kommer inte att närmare belysas. Den komparativa analysen sker med USA som jämförelseland. Detta beror främst på att mängden avgöranden inom området i särklass har sitt ursprung från USA men även på möjligheten att finna lättillgängliga avgöranden.

1.5 Tack

Jag vill rikta ett tack till min handledare Hans Henrik Lidgard för goda råd och vägledning. Ett stor tack särskilt till Johan Förander som ledde in mig på rätt spår när jag kände mig som mest villrådig. Utan dina kloka synpunkter hade resultatet av denna uppsats inte blivit detsamma. Min kära mor tackar jag speciellt för noggrann korrekturläsning under några heta dagar i maj. Slutligen ett tack till Jenna Jameson- tack för att du fanns under arbetets gång.

2. Teknisk inledning

Begreppet datorprogram finns inte definierat i UrL och det juridiska begreppet bygger på en teknisk definition. I vanligt språkbruk menar man med ett datorprogram ”instruktioner för vad en dator skall göra”.[1] För att en dator skall kunna utföra de instruktioner den ges, måste instruktionerna vara givna i en bestämd följd, annars ”förstår” datorn inte innebörden av instruktionerna. Det språk en dator använder sig av är representerat i maskinläsbar form, sk objektkod. Objektkoden är representerad i binär form, dvs ett språk uppbyggt av ettor och nollor. Kort kan sägas att en etta kan representera kommandot ”på” medan en nolla kan representera ”av”. Att skriva ett program i objektkod är ytterst tidskrävande och det har därför utvecklats ett antal språk som representerar de binära instruktionerna för att underlätta programmeringen. Koden språken är representerade i kallas för källkod. Dessa språk brukar delas in i olika nivåer. Ju högre nivå, desto fler binära kommandon representeras av ett kommando i källkoden. Ju högre nivå språket representerar, desto mer användarvänligt är det således. De språk som används mest idag brukar kallas för fjärdegenerationsspråk, t ex Pascal och C++, men utvecklingen går stadigt framåt och femtegenerationsspråken är under utveckling. För att översätta källkoden till objektkod, används ett speciellt program, kallat kompilator eller interpreter. Skillnaden mellan dessa är att en kompilator översätter källkoden till en bestående objektkod, medan en interpreter endast översätter den del av källkoden som skall användas och någon bestående objektkod lagras inte. De som ligger till grund för ett datorprogram brukar kallas för algoritm. Det är svårt att definiera begreppet klart men ett försök till förklaring kan vara att det är algoritmen som kläs i kodtext, dvs kodtexten definierar metoden att lösa programmets funktioner. Algoritmen kan vara en matematisk formel, men behöver inte vara det, vilken resulterar i en logisk följd av funktioner som sedan kan uttryckas i en strukturkarta (se nedan).[2] Algoritmen ses därför som en metod att lösa ett specifikt problem och är därför som sådan inte skyddad upphovsrättsligt.

2.1 Reverse engineering- omvänd utveckling

Vid samtliga nedanstående exempel ligger det en naturlig begränsning i form av i vilken version programmet levereras och i vilken omfattning så kallad omvänd utveckling får ske. I dag är det, med undantag, vanligast att programmen levereras i objektversion. Vid den omvända utvecklingen skall objektversionen av programmet översättas till en källkodsversion. Detta är en process som inte kan ske automatiskt och är mycket tidsödande och kostsam. Man tar utgångspunkt i programmets objektkod och ”bryter ned” koden i instruktioner som sedan sammanställs i programmets funktioner.[3]

[1] SOU 1985:51 s86
[2] Andersen Mads Bryde, Laerobog i EDB, s138f.
[3] Wagle Mediaas Anders, Ödegaard jr Magnus, Opphavsrett i en digital verden, s267.

Det existerar program som kan utföra rekompileringen men resultatet blir inte den ursprungliga källkoden, utan något som liknar denna. För att få den ursprungliga källkoden måste en programmeringskunnig person bearbeta resultatet. Någon källkodsversion existerar inte i detta läge, utan därefter måste en omprogrammering ske i källkod med utgångspunkt i de funktioner som nedbrytningen av objektkoden har genererat. Denna typ av omvänd utveckling är specialreglerad i UrL 26h §. Regeln är avsedd att underlätta för konsumenter att få reda på de principer som ligger till grund för ett program för att möjlighet att kommunicera med andra program skall kunna ske. Olika mjukvaru- och hårdvarutillverkare använder sig av olika standarder och kommunikation kan inte alltid ske. Regeln ställer dock upp vissa kriterier för att denna typ av omvänd utveckling får ske. Det är endast den rättmätige programvaruinnehavaren som får utföra denna utveckling och dessutom endast under just nämnda förutsättning att det sker för att skapa kompatibilitet, med villkoret att dessa upplysningar inte funnits tillgängliga för programinnehavaren.

3. Upphovsrättens grundbegrepp.

Upphovsrätten skall ses som ett efterbildningsskydd som uppkommer formlöst, dvs utan någon form av registrering. Det ligger således i begreppets natur att någon prioritetsrätt inte förekommer, utan skyddet omfattar senare verks kopiering av originalverket, såvida påverkan av originalverket kan styrkas. Skyddet omfattar därför inte att senare verk råkat bli lika originalverket.
I svensk rätt är upphovsrätten reglerad genom lagen om upphovsrätt till litterära och konstnärliga verk (Lag 1960:729), UrL, men lagen har under åren blivit utsatt för åtskilliga förändringar. Bl a har lagen i och med anslutningen till EU, fått anpassas efter de direktiv som trätt i kraft. De viktigaste internationella konventionerna inom upphovsrätten är Bern- samt Romkonventionen. Datorprogrammen regleras sedan 1993 av direktivet 91/250/EEG, efter vilket den svenska lagstiftningen har anpassats.
Upphovsrätten ger upphovsmannen två rättigheter- de ekonomiska och de ideella. Syftet med de ekonomiska rättigheterna är att skaparen till ett verk skall kunna bära frukt av det andliga skapande han har givit uttryck för och de ideella rättigheterna är främst till för att upphovsmannen skall bli namngiven vid utnyttjande av verket.

3.1 Verkskategorierna

Enligt 1 kap 1§ 1 st UrL, åtnjuter skaparen till ett litterärt eller konstnärligt verk upphovsrätt för sin skapelse, oavsett uttrycksform. De litterära och konstnärliga verken är uppdelade i sju undergrupper, varav datorprogram hör till de litterära verken. För att ett verk skall skyddas upphovsrättsligt, krävs det att verket uppnår en viss kvalitet. Det är inget kvalitetskrav i estetisk bemärkelse utan även ”dåliga” verk åtnjuter skydd. Termen som används är verkshöjd, vilket innebär att verket skall ha en viss särprägel. Inom samtliga verkskategorier är det svårt att sätta några generella normer för när ett verk är skyddat, utan bedömningarna överlämnas till domstolarna, där frågan främst kommer upp i samband med intrångsprocesser. Bakom resonemanget om verkshöjd, döljer sig samhällsekonomiska argument och avvägningen är svår att göra. Ett för högt krav på verkshöjden, kan få konsekvensen att det andliga skapandet hämmas, samtidigt som ett för lågt krav kan leda till monopolställning. Detta kan dock i viss mån undvikas genom att kravet ställs lågt, samtidigt som skyddsomfånget begränsas. Problemet med en sådan lösning är dock att tvisterna kan bli många och samtidigt svåra att förutsäga.[4]

[4] Koktvedgaard Mogens, Levin Marianne, Lärobok i Immaterialrätt, s73.

3.2 Bearbetningar.

Bearbetningar regleras i 4§ 1 st UrL. En bearbetning innebär att någon har bidragit med en personlig särpräglad instats av ett existerande verk. Skaparen av bearbetningen åtnjuter upphovsrätt men hans rätt att förfoga över verket är beroende av originalskaparen och förutsätter tillåtelse från denne. Det uppkommer problem vid bedömningen om en bearbetning eller ett nytt verk har uppkommit. Enligt 4§ 2 st UrL är upphovsmannen till ett verk som har åstadkommits i fri anslutning till originalverket inte beroende av rätten till originalverket. Gränsdragningsproblem kan förekomma och även i dessa fall finns inga generella bedömningsmallar att tillgå, utan varje eventuell tvist får bedömas för sig.

4. Kategoriseringen av datorprogram.

4.1 Inledning.

Att datorprogram skyddas upphovsrättsligt råder det idag ingen tvekan om. I UrL 1§ är datorprogram specificerat som en verkskategori som skyddas under litterära verk. När datorprogram även gjorde sitt intåg på den juridiska arenan på 1960- talet inleddes en diskussion inom doktrinen huruvida datorprogram kunde åtnjuta upphovsrättsligt skydd. Jag hade tänkt att redogöra för hur diskussionens vågor rörde sig, även om den idag mer har rättshistorisk karaktär. Detta beror dels på att datorprogrammens intåg i viss mån satte UrLs tillämpning på nya verkskategorier på prov, dels på att en insikt i de resonemang som fördes kan ge en djupare förståelse för datorprogrammens uppbyggnad, vilket kan underlätta för förståelsen av uppsatsens problemställning. Jag ämnar först utreda de teoretiska argument som har anförts i debatten och därefter redogöra för den praxis som har berört frågeställningen.

4.1.1 ”The phantom problem”

Det första som skiljer datorprogram från traditionella verk är mottagaren. Ett datorprogram kan vara fysiskt lagrat i en dator eller annat lagringsmedium såsom diskett eller CD-skiva. Visserligen kan koden även presenteras i form av en pappersutskrift men det är sällan i den formen som den tänkta mottagaren är avsedd att ta del av datorprogrammet. Mottagaren för själva programkoden är datorn, som bearbetar programmets källod och översätter den till objektkod och en möjlig presentationsform är framställningen på datorns skärmbild. Det är dock ofta så, i varje fall när det gäller exempelvis styrprogram för maskiner, att någon ”yttre” presentation överhuvudtaget inte förekommer. Något litterärt verk av beskrivande art i traditionell mening rör det sig inte om, utan instruktioner till en dator.[5] Problemet har i USA kallats för ett ”phantom problem”, dvs det litterära verket är inte riktat till människor utan till en dator.[6] Även om mottagaren är en datormaskin består programmet, både käll- och objektkoden, av de karaktäristiska moment som kännetecknar litterära verk. Ordet litterär härstammar från grekiskans littera (= bokstav) och ett datorprogram består av bokstäver, siffror och symboler, vilket således bör kvalificera datorprogrammen som litterära verk i den bemärkelsen.

[5] SOU 1985:51, s90
[6] Stensaasen Tarjei, Rettslig vern av EDB- programmer og databaser, s43.

4.1.2 Kravet på estetik.

För att skyddas av UrL ställs kravet på särpräglad insats. Det har dock framförts åsikter om andra krav vid sidan om detta. De verk som traditionellt har skyddats av upphovsrätten har ofta haft prägeln av kulturbärare. De är ägnade att bibringa mottagaren en upplevelse, av typisk estetisk karaktär. Knoph kategoriserade upphovsrättens område i vid mening som det estetiska området, med slutsatsen att skydd för verk aktualiserades endast för ”skapande åndsvirksomhet”, dvs för andligt skapande verksamhet.[7] Uppfinningar hörde däremot hemma inom det tekniska området och därmed hänvisade till patentförfarandet. Fransmannen A. Troller har fortfarande den uppfattningen att termen ”aesthetic works” är att föredra framför litterära och konstnärliga verk, detta med anledning av att de skyddsvärda verken direkt refererar till ”our sense of beauty”.[8] Detta hör samman med att de estetiska verken har egenskapen att framkalla omedelbara sinnesintryck, relaterade till en skönhetsupplevelse, hos betraktaren.[9] I USA skyddades tidigare inte verk av icke- dekorativ art med funktionell karaktär, vilket bl a resulterade i att pianorullar avsedda för ett självspelande piano, inte kunde åtnjuta upphovsrättsligt skydd.[10] Det finns dock betydligt senare exempel på domstolars motvilja att skydda datorprogram upphovsrättsligt. En tysk domstol i Mannheim ansåg 1981 att datorprogram inte kunde åtnjuta skydd p g a bristande intellektuellt- estetiskt innehåll.[11] Apellationsdomstolen hade dock en annan uppfattning men fallet tyder trots allt på domstolarnas tidiga osäkerhet i kategoriseringen av datorprogram. En invändning av detta slag idag skulle inte vinna bifall i många läger. Till att börja med är det inte endast datorprogrammen som skulle drabbas av detta ytterligare kriterium för skyddsbarhet. Det är stor spännvidd mellan de litterära verk som åtnjuter skydd idag, allt från poesi och prosa till bruksanvisningar och kartor, så länge verken präglas av en individuell skapande insats. Ett estetiskt krav utöver dessa, skulle leda till att domstolarna måste ta ställning till huruvida verken i tillräcklig utsträckning påverkar vår skönhetsupplevelse inför ett verk, en godtycklig princip som skulle vara ytterst svår att upprätthålla. Precis som en litteraturvetare känner sig känslomässigt berörd av en dikt, finns det programmerare som anser att vissa snillrika lösningar i ett datorprogram är ”vackra”, ”Programs have, like all scientific works, their own aesthetic…”.[12]

4.1.3 Intellektuellt skapande?

Det naturliga språket ger skaparen möjlighet till ett otal kombinationer vid skapandet av ett litterärt verk. Visserligen finns det ett marknadsmässigt intresse för skaparen. Författar någon en tjock tråkig bok lär den inte bli

[7] Seipel Peter, Datorprogrammens rättsskydd, NIR 1973 s143.
[8] Ulmer E, Kolle G, Copyright Protection of Computer Programs, IIC No. 2 1983 s169f.
[9] Seipel Peter, NIR 1973 s146.
[10] 201 US (1908), White- Smith Music Pub. Co v. Apollo Co.
[11] Inkasso- domen, Bundesgerichthof I ZR 52/83 850509. IIC 1983 s168.
[12] Ulmer E, Kolle G, No 2. 1983 s172 not 41.

såld, precis som en ful vas inte lär göra någon succé på marknaden. Detta intresse finns givetvis även för skaparen av ett datorprogram- ett dåligt program med många fel och som upptar stort minne, har inga större möjligheter på marknaden. Den stora skillnaden mellan det ”naturliga” språket och programmeringsspråket, är dock att det senare är bundet av funktionella regler. Ett felaktigt kommando innebär att den avsedda effekten uteblir då datorn inte ”förstår” vad som åsyftas. En spridd uppfattning är att denna funktionsbundenhet innebär att utrymmet för det individuella spektret är mer begränsat för utformningen av ett datorprogram än vid annan litterär text, dvs den logiska strukturen i programmeringsspråket begränsar möjligheterna till personlig utformning.[13] Detta skulle i sin tur även påverka nivån på verkshöjden, vilket närmare kommer att behandlas nedan, men det är en fråga man får ta ställning till efter det att utredning har gjorts för verksbestämningen.

I EG- direktivet för skydd av datorprogram nämns i Art 1.3, angående skyddsbarheten, att datorprogram skyddas om ”det är originellt i den meningen att det är upphovsmannens egen intellektuella skapelse.” Att ett datorprogram är resultatet av intellektuellt arbete är idag kanske enligt de flesta en självklarhet. Frågan är i dock att vid en närmare anblick bestämma vad det är i datorprogrammen som motsvarar det intellektuella. Det intellektuella torde inte ligga i arbetsinsatsen, då skyddet inte är ett konkurrensrättsligt sådant utan ämnar att skydda andligt skapande. Ett program skall vara användarvänligt, billigt och effektivt på samma gång som minnesutrymmet skall begränsas i så hög utsträckning som möjligt.[14] Det är således ett otal faktorer som skall korrelera för att programmet skall anses fullvärdigt. Detta är något som präglar hela utvecklingsprocessen för ett datorprogram, från utvecklingen av algoritmen till designstadiet. Även själva programmeringen präglas av intellektuellt arbete och är i de flesta fall inte resultatet av rent mekaniskt arbete. Den logiska bundenheten är dock inte så stark att det intellektuella arbetet inskränks i så stor utsträckning att det inte finns något utrymme för originalitet. Den intellektuella prestationen är därför inte mindre än vid skapandet av traditionella litterära verk .

4.2 Finns det utrymme för originalitet, personlighet?

Efter att ha konstaterat att ett datorprogram är produkten av ett intellektuellt arbete, måste man i därefter konstatera att det dessutom finns utrymme för personlighet och originalitet. Detta brukar vid andra verkskategorier kategoriseras som särprägel.[15] Frågan är om datorprogrammens logiska struktur innebär att utrymmet för skaparens personlighet inskränks eller till och med sätter fullständigt hinder för upphovsrättsligt skydd och således hänvisas till andra skyddsformer. Denna fråga var i datorprogrammens

[13] Jfr Schmidt Per Håkon, Teknologi og immaterialret, s211f.
[14] Stensaasen Tarjei, s175.
[15] Koktvedgaard Mogens, Levin Marianne, s73.

initialskede inom juridiken främst en fråga som behandlades av rättsvetenskapsmän. I början på 1980- talet, när de första målen som rörde kränkning av datorprogram nådde domstolarna, fick frågan synnerligen stor betydelse. De rättsvetenskapsmän som starkast argumenterade för uteslutet skydd för datorprogram, var tyskarna Sidler, Kummer och Köhler samt fransmannen Troller.

4.2.1 Troller och Sidler.

Troller anser att datorprogram är uppbyggda av strängt logiska och tekniska regler som inte lämnar något utrymme för originalitet.[16] Således är det programmens natur som innebär ett hinder för skydd. Troller anser att originalitetskravet inom upphovsrätten är ett strängt unikumvillkor, vilket innebär att det skyddsvärda i en prestation skall vara unik för det presterade. Både Sidler och Troller anser att den logiska strukturen och strävan efter perfektionism i programmet resulterar i en strävan efter ett befintligt ”idealprogram”.[17] Programmeringen präglas av denna strävan att finna idealprogrammet, det tekniskt bästa programmet som löser de uppgifter det är avsett att göra. Sidler menar visserligen att det förekommer olika vägar att lösa ett problem med programmering och anser även att det finns ett brett spektra mellan ideallösningen och de ”obrukbara” (sic!) lösningar som en programmeringsuppgift kan resultera i.[18] För att stödja sin teori utgår Sidler från ett hypotetiskt scenario. Om ett godtyckligt antal programmerare försöker lösa samma programmeringsuppgift under obegränsad tid, resulterar detta i att samtliga lösningar kommer att vara identiska, den ideala lösningen på uppgiften har uppnåtts. Slutsatsen, enligt både Troller och Sidler, är därför att i själva strukturen i programmeringsspråket ligger en bundenhet och en strävan efter den optimala lösningen som i sig utesluter upphovsrättsligt skydd. Kummer, som delar Sidlers och Trollers uppfattning, anser att för att skydd skall åtnjutas krävs att prestationen skall vara resultatet av ett arbete präglat av ändamålsfrihet, ”zweckfreiheit”.[19] Ändamålsfrihet vid utformningen av ett verk kan endast föreligga när resultatet av arbetet kan ges ett stort antal utformningar, utan att uppnåendet av ändamålet påverkas. En kursbok i immaterialrätt kan tjäna som jämförelse. Ändamålets uppnående är att skriva en bok som är lättförståelig för studenter med knappa kunskaper inom ämnesområdet. Författaren är visserligen bunden av de objektiva fakta som föreligger men kan trots detta utforma boken på ett flertal olika sätt utan att grundkriteriet, att skriva en bok för studenter, påverkas. Ändamålsfriheten är därför så stor att skydd för de olika utformningarna är motiverade. Kummer menar att detta krav inte är uppfyllt för datorprogram trots att han, något överraskande, anser att någon möjlighet, inte ens teoretisk, till ett idealprogram inte föreligger. Det som hindrar upphovsrättsligt skydd är dock att ändamålsfriheten är för liten. Även Köhler anser att totaloptimerade program existerar men att de dock

[16] Seipel Peter, NIR 1973 s145.
[17] Seipel Peter, NIR 1973 s145.
[18] Seipel Peter, NIR 1973 s145.
[19] Seipel Peter, NIR 1973 s146.

endast har ett teoretiskt intresse. Köhler definierar dessa program som den bästa tekniska lösningen utifrån en given situation.[20] Till skillnad från ovanstående resonemang, anser dock Köhler att programskaparen på ett tidigt stadium ställs inför valmöjligheter i sitt skapande. Dessa valmöjligheter ger således utrymme för programmeraren att ge verket en personlig prägel och utrymme för originalitet öppnar sig.

4.2.2 Motargument.

En författare av ett traditionellt litterärt verk har obegränsade möjligheter att uttrycka sig i skrift. Det som begränsar hans uttrycksmöjligheter är mer marknadens krav. En tjock, ointressant bok med meningslösa ordkombinationer får antagligen svårt att finna någon köpare. En skapare av ett datorprogram är givetvis även han bunden av att skriva ett bra program avsett att attrahera köpare men han är även bunden av de formella regler som programmeringsspråket är anpassat efter. Det finns dock exempel på traditionella litterära verk där även denna bundenhet förekommer, som trots detta åtnjuter skydd. Det gavs ovan ett exempel på en läroboksförfattare som är bunden av själva ämnesvalet och de objektiva fakta som föreligger. En utformning av en bruksanvisning kan tjäna som ett mer klargörande exempel på ännu högre grad av bundenhet. Variationsvidden för utformningen är så pass stor att det finns utrymme för personlig prägel och originalitet. Detta trots en klar bundenhet av de snäva gränser som ämnesvalet sätter. Ett datorprogram består ofta av väldigt många instruktioner till datorn, programmen som sådana är kanske 100 gånger större än de instruktionsböcker som medföljer programmet och som visar dess funktioner. Till skillnad från Sidler och Kummers resonemang ovan, är ett program ofta inte utformat att lösa en uppgift, utan ett flertal. Om utgångspunkt tas i ett modernt ordbehandlingsprogram, är det givetvis skaparens uppgift att utveckla ett program som är användbart för att skriva löpande text. Själva uppgiften är däremot inte slutförd i och med att detta, utan ett otal ytterligare funktioner är knutna till programmet. Texten skall kunna redigeras, korrigeras och skrivas ut, samtidigt som designen inte är en obetydlig faktor. Den optimala lösningen är således beroende av ett flertal faktorer som utan inbördes rangordning spelar en väsentlig roll. För att en optimal lösning skall existera måste således programmet innehålla två faktorer, dels måste det bygga på en uppgift att lösa, dels måste denna uppgift vara av ytterst trivial natur, så att utrymmet för det som Kummer betecknar ändamålsfrihet är begränsat. Bristen på ändamålsfrihet är således inte ett självuppfyllande kriterium som generellt beskriver verkligheten i programmeringsmiljön, utan drabbar endast en liten del av de program som skapas och som dessutom har ett obetydligt kommersiellt intresse.

Ulmer och Kolle är mer kategoriska i sin ståndpunkt angående optimala program och menar att optimala program ”is a fiction remote from practice”.[21] De refererar till ett experiment som gjordes bland ett antal

[20] Seipel Peter, NIR 1973 s147f.
[21] Ulmer E, Kolle G, IIC No 2. 1983 s177f.

studenter. De försågs med samma programmeringsuppgift och blev ombedda att lösa den på det mest effektiva sättet. Det visade sig att resultaten varierade kraftigt, utan att någon av lösningarna kunde anses som den mest effektiva eller optimala. Ett effektivt program i ett avseende kunde få sin motsvarighet i en annorlunda utformning. Även Seipel anser att teorin om totaloptimerade program är oriktig. I en programmeringsmiljö är det inte möjligt att kunna urskilja den bästa lösningen då optimeringsmöjligheterna är alldeles för vaga och motsägelsefulla.[22] Seipel menar att Sidlers och Kummers resonemang bygger på felaktiga förställningar om datorprogrammering och efterlyser i sin artikel ett mer designinriktat tänkande rörande datorprogram och vill därmed avsluta debatten om datorprogram som enbart härrörande till den tekniska sidan av den rättsliga kategoriseringen. Detta är ett faktum som även påverkar bestämningen av verkshöjden som behandlas nedan.

4.3 Utrymmet för originalitet.

Utrymmet för personlighet och originalitet yttrar sig redan på ett tidigt stadium i utvecklingen av ett program och avser både innehållet och formen av programmet.

4.3.1 Innehållet.[23]

Ursprunget till själva lösningarna på de uppgifter man vill att ett program skall utföra grundar sig i algoritmen i programmet. Detta är själva idéerna bakom programmet, dvs det oskyddade innehållet. Algoritmutformningen tar sig uttryck i rutiner och subrutiner som är länkade till varandra. Vid större program uttrycks det i komplicerade strukturkartor som konkretiserar det okodade programmet. Redan på detta stadium finns det stora variationsmöjligheter för programmeraren att ge uttryck för sin kreativitet och personlighet. Det går redan här att säga att möjligheterna till utformning inte är knuten till någon optimal programmeringsform, utan är anpassad efter den metod och det tycke som ligger närmast tillhands för den enskilde programmeraren. Strukturkartorna är resultatet av de val programmeraren gör av in- och utmatningsmoment i programmet vilka är beroende av varandra. Dessa bygger på den grundläggande algoritmen men denna kan uttryckas på flera olika sätt, beroende på vilken metod man använder. När detta är gjort skall dessa struktureras så att de är förenade på ett sätt som uppfyller funktionen av programmet. Här ligger en stor del av själva programdesignen och den resultatet av programmerares personliga kreativitet och sätter den personliga prägeln på programmet.

[22] Seipel Peter, NIR 1973 s148.
[23] Ulmer E, Kolle G, IIC No 2. 1983 s178.

4.3.2 Överföring till programkod.[24]

Efter detta mycket omfattande arbete som är själva grundstrukturen på programmet, skall det överföras till källkod som datorn sedan översätter till objektkod. Även på detta stadium finns det stora möjligheter för personlighet och originalitet från programmeraren. Det programspråk han väljer, är ofta bestämt på ett tidigare stadium vid skapandet av strukturkartorna. När kodningen av programmet skall ske kan skaparen ofta välja olika alternativ för att få den avsedda funktionen. Exempelvis kan anropandet av en procedur utformas på många sätt utan att man kan påstå att ett optimalt sätt finns. Med tanke på ett programs komplexitet och många alternativa programmeringsvägar, måste man påstå att sammantaget finns det stora möjligheter för originalitet även i själva kodningen av programmet.

4.4 Praxis.

4.4.1 Hovrätten för västra Sverige 19/11 1987.

I svensk praxis förekommer det mycket få rättsfall som rör datorprogram och upphovsrätt. Innan lagändringen 1989, då datorprogram infördes som en skyddad kategori under litterära verk i 1§ UrL, finns inget vägledande avgörande från högsta domstolen. Frågan angående kategoriseringen på skydd uppkom dock i ett hovrättsavgörande.[25] Fallet rörde piratkopiering av datorprogram, där åklagaren yrkade ansvar enligt lagen om upphovsrätt till litterära och konstnärliga verk. Den tilltalade bestred ansvar enligt nämnda lag. Även om fallet inte har status av prejudikat, är det intressant då åklagaren i hovrätten åberopade ett yttrande av professor Peter Seipel. Tingsrätten konstaterade att den icke uttömmande första paragrafen i UrL, är avsedd att skydda beskrivande framställningar ”oavsett det sätt varpå den kommit till uttryck”[26]. Domstolen konstaterar att datorprogram kategoriseras som en serie anvisningar till en dator att utföra en viss uppgift. Det som tingsrätten dock tar fasta på, är att produkten skall vara ett uttryck för andligt skapande och skall ha prägel av viss självständighet och originalitet. Med detta vill domstolen utesluta verksamhet av rent mekaniskt arbete. Tingsrätten anser visserligen att datorprogram kan åtnjuta skydd, detta med beroende av lagens vida tillämpningsområde, men utifrån rättens formuleringar är det med viss tvekan. Rätten skriver att datorprogram i allmänhet är av karaktären tekniska hjälpmedel för att nå ett resultat och ”mindre utgöra produkter av andligt skapande /---/ och framstå som uttryck för dennes [upphovsmannens] individualitet”.[27] Hovrätten hänvisar i sin dom till utredningen som föregicks av lagändringen 1989, att program av

[24] Ulmer E, Kolle G, IIC No 2. 1983 s177.
[25] HovR f. Västra Sverige 19/11 1987. Publicerat i NIR 1988 s310.
[26] HovR f. Västra Sverige 19/11 1987, NIR 1988 s313.
[27] HovR f. Västra Sverige 19/11 1987, NIR 1988 s314.

inte för enkel art ”torde åtskilliga valmöjligheter finnas för utformaren”[28] och får även stöd av professor Seipel.

4.4.2 Utländsk praxis

Från utlandet finns två rättsfall som mycket tydligt exemplifierar den osäkerhet som präglade domstolarna om datorprogrammens eventuella upphovsrättsliga skydd. Det ena är från Australien[29] och det andra är från Frankrike[30]. Det intressanta med dessa fall är, att de rör samma program men domstolarna i de båda länderna kom till olika slutsatser. I det australiska ansåg domstolen att datorprogram inte kan vara av litterär karaktär enligt lagens mening, utan att de endast syftar till att assistera en maskin. Domstolen stödde sig visserligen på att frågan varit uppe vid en lagändring, då filmer ansågs åtnjuta litterär karaktär men ej datorprogram. Utfallet resulterade i kraftig reaktion från myndigheter som krävde en utredning för att ge datorprogram upphovsrättsligt skydd.[31] Den franska domstolen kom till motsatt resultat. De stödde sig på ett banbrytande fall från 1982, där Cour d´Appel de Paris, ansåg att ett datorprogram ”är ett alster av andligt skapande, originellt till sammansättning och uttryck /---/ och att det inte är fråga om en nödvändig intellektuell mekanism”.[32] Den franska domstolen berör visserligen utrymmet för personlighet medan den australiska motsvarigheten kategoriskt avfärdar datorprogram som litterära verk. Det intressanta ligger dock i att två domstolar vid samma tidpunkt kommer fram till helt skilda slutsatser.

[28] HovR f. Västra Sverige 19/11 1987, NIR 1988 s319.
[29] Apple Computer Inc. and Apple Computer Australia Pty Ltd v. Computer Edge Ltd and Michael Suss. Fed Ct. of Australia, NSW Distr. Reg. Gen. Div. 7/12 1983. Karnell NIR 1984 s211.
[30] Apple II v. Golem, Trib. Gr. Inst. Paris 14/6 1983. Karnell NIR 1984 s216.
[31] Karnell, Upphovsrätt till datorprogram, NIR 1984 s211 not 19.
[32] Soc. Babolat Mailott Witt c. P., Gaz Pal. 3/3 1983 s117. NIR 1984 s212.

5. Vad är skyddat?

5.1 Inledning

Då datorprogram i viss mån är specialreglerad i UrL, är definitionen på datorprogram viktig att ha klar för sig. I förarbetena till ändringen av UrL, konstaterades att det fanns en risk med en precis definition på begreppet, då den tekniska utvecklingen på området kunde göra definitionen förlegad.[33] I WIPO:s modellbestämmelser för datorprogram utarbetades en definition där datorprogram beskrivs som ”…en serie av instruktioner eller anvisningar, oberoende av den uttrycksform eller den anordning vari den är nedlagd, avsedd att förmå en dator att direkt eller indirekt ange eller utföra en speciell funktion eller uppgift eller uppnå ett speciell resultat”. Klart var att både programmets objekt- och källkod skyddades som ett verk, oavsett i vilken form det lagras eller är nedskrivet på papper. Objektkoden ses alltså inte som ett exemplar av källkoden utan är att hänföra till samma verk. Även skärmbilden som genereras av koden skyddas som ett datorprogram. Detta är viktigt ur den aspekten att originalitetsbedömningen måste göras utifrån alla dess framträdelseformer, då EG- direktivets artikel 1.2 nämner att datorprogrammet skyddas i alla dess uttrycksformer. Ett tänkbar, men dock sällsynt, fall är att skärmbilden som genererar koden är originell, men koden inte är det, skyddas även den bakomliggande koden.[34]

5.2 Förberedande designmaterial.

Med EG- direktivets ikraftträdande kom även sk ”prepartory design work” att omfattas av begreppet datorprogram, i den omfattningen att de leder till utvecklandet av ett datorprogram och att resultattet blir ett datorprogram.[35] Det står inte angivet i vilken utsträckning designmaterialet måste vara specificerat. Mycket talar dock för att det i varje fall måste ange tekniska lösningar för programmet.[36] Även om det förberedande designmaterialet kategoriseras som datorprogram, kommer detta att räknas som ett nytt verk och inte under samma verk som käll- och objektkoden gör. Det är viktigt att skilja på sk ”supporting material”, dvs material som behövs för att kunna använda programmet, jfr teknisk litteratur. Givetvis kan detta material skyddas som ett traditionellt litterärt verk, såvida kravet på verkshöjd är uppfyllt.

[33] SOU 1985:51 s86.
[34] Bender Hanne, Ophavsret til brugergraenseflader ”Look and Feel”, NIR 1997s376.
[35] Prop. 1992/93 s112.
[36] Bing Jon, Opphavsret og ny informasjonsteknologi: Noen spredte notater, NIR 1995 s604.

5.3 Oskyddade element.

Precis som för andra delar av upphovsrätten, finns det vissa element som inte kan åtnjuta skydd. Bakomliggande idéer och principer för ett program kan aldrig skyddas. Detta har givetvis att göra med att risken för monopolställning hade varit stor. Nedan i kapitlet om plagiatvärderingen kommer jag att redogöra för att det ibland kan vara svårt att skilja idéerna från utformningen av verket. EG- direktivet är svårtolkat i avseendet vad som kan åtnjuta skydd som datorprogram. Nedan kommer jag i avsnittet om plagiatvärderingen, redogöra för i vilken omfattning de icke- litterära aspekterna i ett program kan åtnjuta skydd, något som inte framgår av förarbetena. Direktivet utesluter visserligen inte skydd för tekniker, algoritmer och programmeringsspråk. En programmeringsteknik, sk pionjärprogram, som är en objektiv nyhet kan därför i vissa fall åtnjuta skydd.[37] Skyddet kan dock endast omfatta utformningen av dessa företeelser.[38] Det föreligger därmed en diskrepans mellan EG- direktivet och de svenska förarbetena. Förarbetena hänvisar endast till programmets objekt- och källkod såsom skyddsobjekt. Det konsekventa hävdandet i svenska förarbetena, att algoritmen i ett program inte kan åtnjuta skydd, har med EG- direktivets ikraftträdande utsatts för kritik.[39] Jag tänker inte gå närmare in på kritiken men den syftar på EG- direktivets generella ställningstagande av skyddsobjektet datorprogram inte är förenlig med uppfattningen i förarbetena. Plogell hävdar att utvecklingen går mot ett mer funktionsinriktat skydd för datorprogrammen. Algoritmerna skiljer sig från abstrakta idéer som inte kan åtnjuta skydd, i den bemärkelsen att algoritmen inte kan frånkännas sin funktionella relevans.

5.3.1 Public Domain- Fria element.[40]

Public domain är det amerikansk uttrycket för verk som kan utnyttjas fritt, även om man skulle kunna förvänta sig att det fanns en rättighet knutet till verket. Inom programvaruindustrin är det inte helt ovanligt att upphovsmannen har sagt ifrån sina ekonomiska rättigheter knutna till verket. På Internet finns det många exempel på sk ”free- warez”, program som är tillåtna att ladda hem men även föra att sprida vidare. Orsaken till detta kan vara många men ofta handlar det om att producenten på ett enkelt sätt vill sprida sina nya produkter. En sk demo- version släpps för fritt utnyttjande, för att locka till sig potentiella köpare av den fullständiga versionen. Andra exempel är programmoduler som i och för sig uppfyller kravet på verkshöjd men som används för att skapa kompatibilitet mellan program och hårdvara i sådan utsträckning att de har blivit allmängods. Den ideella rätten upphör dock inte med att verket blir ett public domain.

[37] Plogell Michael, Immaterialrättsliga aspekter på datorprogram, s22.
[38] Prop. 1992/93:48 s113.
[39] Plogell Michael, s31ff.
[40] Wagle/Ödegaard, s134.

6. Verkshöjdsbedömningen

6.1 Inledning

Precis som för andra verkstyper, finns det för datorprogram ett krav på att verkshöjd skall föreligga för att verket skall skyddas. Att klart fastställa vad denna gräns går utifrån objektiva kriterier är givetvis omöjligt, utan vid tvist överlämnas denna bedömning till domstolarna. Vid intrångsprocesser vid datorprogram är verkshöjdsgränsen mest intressant utifrån två perspektiv. Det första är då motparten anser att det påstådda kränkta verket inte har verkshöjd och det andra är vid situationer när delar från kärandes program infogats i motpartens verk. Vid en sådan tvist måste gränsen mellan ett ”fritt element”[41] och verket bestämmas. När man talar om verkshöjden för datorprogram, är det viktigt att konstatera vad man syftar på. Nedan kommer jag att redogöra för själva *programkoden*, dvs objekt- och källkoden. Detta måste klart skiljas från de utkomponenter som programmet kan generera i form av grafik, skärmtexter, ljud, funktioner etc.

Diskussionen om verkshöjd tenderar ofta att bli väldigt akademisk men är principiellt viktig. På andra sidan av samma mynt befinner sig verkets skyddsomfång. Ett för lågt krav riskerar att närma sig idéskydd, vilket i sin tur kan resultera i att monopolställning etableras hos rättighetsinnehavare, med följden att den teknologiska utvecklingen bromsas upp. Om skyddsomfånget dessutom är väldigt snävt finns risken att domstolarna belastas med intrångsprocesser. Ett för högt krav å andra sidan, kan resultera i att välutvecklade program inte åtnjuter skydd, trots kraftiga arbetsinsatser, som i sin tur även det kan resultera i att utvecklingen bromsas upp, då riskkapital inte satsas i tillräcklig utsträckning. Skyddsomfånget blir förvisso större och intrångsprocesser inte lika förekommande. Avvägningen är således beroende av många faktorer.

6.2 Bedömningsfaktorer

Verkshöjdskravet för datorprogram diskuterades i samband med lagändringen 1989.[42] I propositionen uttalas det att verkshöjdskravet ”bör ställas förhållandevis högt”.[43] En hjälpnorm för detta kriterium är att om det finns risk för dubbelskapande, dvs om två personer oberoende av varandra skapar samma program, skall verkshöjdskravet inte anses uppfyllt. Utrymmet för originalitet är i så fall så snävt att skydd inte kan tillåtas.[44] Efter lagändringen har vi dock fått en viktigare tolkningskälla i form av EG-

[41] Termen är hämtad från Ödegaard/Wagle. Jfr ”public domain”.
[42] Se SOU 1985:51 och prop 1988/89:85.
[43] Prop 1988/89:85 s27.
[44] Vidare om Dubbelskapande se Nordell Per Jonas, Dubbelskapande i teori och praktik, NIR 1995 s630.

direktivet för skydd för datorprogram, 91/250/EEC. Artikel 1.3 i direktivet uttrycker att ”Ett datorprogram skall skyddas att det är originellt i den meningen att det är upphovsmannens egen intellektuella skapelse. Inga andra bedömningsgrunder skall tillämpas”. I preambeln till direktivet nämns datorprogrammets kvalitativa och estetiska egenskaper som faktorer som inte påverkar skyddsbarheten. Avsikten med direktivet är givetvis att harmonisera medlemsstaternas verkshöjdskrav. Det land som främst fick omvärdera sitt verkshöjdskrav, var Tyskland. I den tidigare omnämnda Inkassodomen, uttalade domstolen att även ett kvalitetskrav skall ligga till grund för skyddsbarheten, ett kvalitetskrav som för tankarna till svensk patentlagstiftning, som innebär att den kvalitativa nivån skall ligga över kunskaperna hos en genomsnittlig fackman. Ordagrant är innebörden av direktivet ett avsteg från det traditionella verkshöjdskravet vi tidigare hade i svensk rätt. Artikeln syftar endast på att skapelsen är originell i subjektiv bemärkelse, dvs något krav på särprägel och intellektuell skapande insats föreligger inte. Således skulle kravet på verkshöjden enligt direktivet endast innebära att upphovsmannen är skyddad mot direkt kopiering, en invändning som gjordes från ett antal remissinstanser inför harmoniseringen av EG-direktivet. Från lagstiftarens sida bemöttes detta argument att det inte ställs några särskilda krav på verkshöjden vad avser datorprogram och andra verkstyper. Direktivet innebär därför inga ändringar p.g.a. direktivets ikraftträdande.[45] Det hävdas även från annat håll att direktivet inte innebär något större avsteg från det svenska verkshöjdsbegreppet.[46] Alltefter som det utarbetas en praxis inom gemenskapen är det dock ordalydelsen av direktivet som är gällande. Idag finns det ingen vägledande europarättslig praxis som berör verkshöjden enligt direktivet.

Lagstiftaren ansåg således att någon ändrad inställning för verkshöjden för datorprogram inte behövde ske i svensk rätt i och med direktivet, i varje fall inte förrän någon gemenskapspraxis bildas på området. Det som är vägledande för verkshöjden är därför fortfarande uttalandet i prop. 1988/89:85, om att kravet bör ställas förhållandevis högt, läst tillsammans med prop. 1992/93:48, att inga särskilda krav på verkshöjden skall ställas på datorprogram jämfört med andra verkstyper.

6.2.1 Kvalitativa hänsyn till trots?

6.2.1.1 Funktionsbundenheten

Som nämns i SOU 1985:51, är det omöjligt, precis som vid andra verkstyper, att objektivt precisera den grad av individualitet och originalitet som krävs för att ett datorprogram skall anses åtnjuta skydd.[47] Risken för dubbelframställning har utvecklats som en hjälpnorm för att precisera denna nedre gräns. Diskussion har dock förts angående verkshöjdskravets påverkan av datorprogrammens funktionsbundenhet. Det har förts fram argument att denna bundenhet måste resultera i att verkshöjden sätts högre för

[45] Prop 1992/93:48 s113.
[46] Rosén Jan, Swedish Software Law, s19
[47] SOU 1985:51 s89.

datorprogram jämfört med traditionella litterära verk.[48] Denna bundenhet kan även finnas, som ovan nämnts, hos de traditionella verken, exempelvis i bruksanvisningar, som ju trots detta kan åtnjuta skydd. Schmidt menar att man inte okritiskt kan jämföra programspråket med det naturliga språket, endast för att de båda består av samma byggstenar, bokstäver. Den logiska funktionen i programspråket ställer högre krav på skaparen att uttrycka sig på ett originellt och individuellt sätt, samtidigt som han dock anser att det finns ytterst stora möjligheter att ge detta uttryck. Schmidt förordar istället att verkshöjdskravet skall närma sig dem som finns inom brukskonsten, detta för att datorprogrammen i sin utformning har stora likheter med brukskonsten i det avseendet att de båda präglas av funktionalitet. Schmidt har fått kritik för sin inställning inom nordisk doktrin, främst från Andersen.[49] Andersen menar att denna typ av argumentation grundar sig på ett missförstånd av strukturen hos programmeringsspråken. Andersen citerar Anthony Clapes rörande juristers bristande kunskaper i informatik såsom ”Most of the commentaters know little about computer programming. They are arguing on the basis of faith rather than knowledge”.[50] Det som skiljer programspråket från det naturliga språket är endast representationsformen. På samma sätt som det hos det naturliga språket finns närmast oändliga möjligheter att uttrycka sig, sätter programmeringsspråket inte några funktionella gränser för att förmedla samma information. Visserligen är man i viss mån bunden till enkla standardkommandon men detta menar Andersen är jämförbart med den bundenhet som finns i val av bokstäver i det naturliga språket. Andersen är dessutom väldigt kritisk till att Schmidt drar likheter med verkshöjdsbedömningen för brukskonst. Han menar att det är att direkt gå emot förarbeten som kategoriserar datorprogrammen som litterära verk. Det finns därför varken reella eller formella grunder för detta. Schmidt kritiserar dock Andersen och menar att han bortser från realiteter (sic!).[51] Förvisso finns det samma möjligheter att uttrycka motsvarande mängd information vid programmering som det finns med det naturliga språket. Programmeringsmiljön sätter dock vissa praktiska och i viss mån ekonomiska hinder för detta. Funktionaliteten lyser även igenom på ytterligare plan, nämligen att programmet skall präglas av effektivitet. Effektiviteten består främst i att åtgången av minne skall begränsas i så stor utsträckning som möjligt.

6.2.1.2 Storlekskriteriet

Diskussionen ovan för tankarna tillbaka på problemen med att applicera det traditionella verkshöjdskravet på datorprogram. Förarbetena nämner att ett program inte får vara enkelt eller okomplicerat. Med tanke på att skyddsobjektet som hänvisas i förarbetena är käll- och objektkoden, är frågan om det är mängden kodtext som avses. Hovrätten nämner i nedan nämnda fall att det aktuella kalkylprogrammet omfattade 250 sidor programkod och verkar till viss del lägga detta till grund för bedömningen

[48] Schmidt Per Håkon, Teknologi og immaterialret s209f.
[49] Andersen Bryde Mads, Laerebog i EDB- ret 1991, s201f
[50] Andersen Bryde Mads, 1991, s119.
[51] Schmidt Per Håkon, Ophavsret og EDB, Vennebog til Mogens Koktvedgaard, s44f.

om utrymme för individualitet finns. I den tyska Inkassodomen nämner domstolen att gränsen för verkshöjd bör ligga på program som innehåller 500-1000 instruktioner. Det givna argumentet är att språk med generationsmässiga skillnader kräver olika omfång, vilket gör att mängden programkod är beroende av vilket språk som används. Detta får definitivt anses vara överspelat i och med rådsdirektivet. Det tyder även på bristande programmeringskunskaper. Som Andersen nämner skulle detta även innebära att större program med mindre ekonomisk programmeringsteknik skulle gynnas framför de program som finner mer ekonomiska vägar att programmera. Originaliteten kan framträda på ett tidigt stadium och skulle därför drabbas av ett storlekskriterium.

6.2.1.3 EG- direktivet

I preambeln till direktivet nämns även att inga kvalitetsaspekter på programmet spelar in i bedömningen av originaliteten. Detta rimmar illa med den programmeringsmiljö som skaparen av ett program måste anpassa sig till. I många fall är just originaliteten i ett program just representerad i programmets kvalitet. Med incitamentet att göra ett program så minnessnålt som möjligt, drivs skaparen av ett program att lösa en uppgift på ett så smidigt och smart sätt som det går. Dessa kvalitetsaspekter i samma avseenden, är inte lika påtagliga inom de andra verkstyperna. Viljan att optimera ett program innebär att programmeraren försöker begränsa antalet variabler men på samma gång göra programmet lika effektivt. Kvalitetsaspekten är därför en inneboende premiss som varje programmerare, som vill skapa ett program med kommersiell framgång, strävar efter. Problemet är att kvalitet och incitament är begrepp som är relaterade till arbetsinsatsen hos skaparen. Upphovsrätten är inget konkurrensrättsligt skydd, med ett visst undantag från katalogregeln i 49§ UrL, utan syftet är att skydda andligt skapande. Den sk ”sweat of the brow”-doktrinen har även försvagats i USA.[52] De grundläggande upphovsrättsliga tankegångarna sätts på spel om ett skydd för arbetsinsatsen premieras. Problemet är återigen att datorprogrammen i stor utsträckning skiljer sig mot de traditionella litterära verken, då kvalitetsaspekten är påtaglig i en större utsträckning. En ren mekanisk arbetsinsats skall aldrig premieras med skydd men kvaliteten kan ha betydelse för bedömningen av skyddsomfånget.[53] Vad åsyftas då med kvalitetsaspekten i originalitetsbedömningen i direktivets preambel? Till skillnad från förarbetena torde okomplicerade program kunna åtnjuta skydd men kvalitetsaspekten har betydelse vid bedömningen av skyddsomfånget.[54]

6.3 Partitionering

Om vi konstaterar att verkshöjdskravet är högre för datorprogram än för litterära verk, innebär det vice versa att kvaliteten som upphovsrättsligt

[52] Feist Publications, Inc. v. Rural Telephone Service Company, Inc. Supreme Court of the United States, No. 89-1909, 27. March 1991.
[53] Plogell Michael, s38.
[54] Plogell Michael, s25f.

skyddat verk går förlorad på ett tidigare stadium för datorprogram än för litterära verk. Som ovan har nämnts måste dock denna bedömning ske individuellt relaterat till varje konkret fall utan att någon teoretisk gräns kan dras. En jämförelse med en roman kan dock vara intressant för att undersöka om det kan dras några generella slutsatser rörande datorprogram och när verkskvaliteten går förlorad.[55] Detta kallas för partitionering, något som får bedömas efter varje konkret fall. En roman på tio kapitel, som ett exempel på ett litterärt verk, uppfyller generellt kravet på verkshöjd. Delas boken upp i två delar, innebär detta att två verk med fem kapitel skapas. En ny uppdelning kan återigen göras osv med resultatet att tio separata verk har bildats. Därefter fortsätter uppdelningen i stycken, meningar och ord, tills ett stort antal delar har uppstått. Trots detta kan det antagligen generellt sägas att romanen, kapitlet och avsnitten kommer att ha verkskvalitet men när uppdelningen har nått meningar och ord, har verkskvaliteten gått förlorad. Om motsvarande uppdelning vidtas för ett datorprogram, kommer uppdelningen resultera i att ur ett helt datorprogram utkristalliserar sig moduler, procedurer, funktionslinjer och slutligen enkla kommandon. Den ovan nämnda funktionsbundenheten, vilken begränsar skaparens förmåga att uttrycka sig originellt vid funktionslinjer, resulterar i att varken dessa eller kommandona kommer att ha verkskvalitet. Wagle/ Ödegaard menar att, till skillnad från romanen som har verkskvalitet vid ett avsnitt, det råder större tveksamhet för ett datorprograms procedur. Min åsikt är att man får uttrycka sig ytterst försiktigt rörande procedurerna. Klart är att de många standardprocedurer och mindre procedurer inte kommer att ha verkskvalitet. Situationen kan vara sådan att en procedur i ett program har ett innehåll som gör att programmet får en viss särprägel, eller är något av en pionjär bland programmeringstekniken.

6.4 Praxis.

6.4.1 BBS- målet.

Som tidigare nämnts är praxis på området mycket begränsad, både avseende gemenskapsrättslig och svensk sådan. Det enda målet som har nått högsta domstolen är det sk BBS- målet.[56] Målet rörde datorprogram som överförts till en BBS (Bulletin Board System). Den tilltalade hävdade att datorprogrammen inte uppnådde verkshöjd, något som åklagaren bestred. Hovrätten menar att för att skydd skall vara för handen, måste det föreligga en viss kvalitet på programmen, dvs de att de inte får vara för triviala i sin utformning eller vara givet av logiska och tekniska aspekter. Trivialitetskravet ansåg domstolen vara passerat med hänvisning till att programmet bestod av en stor mängd instruktioner. Det faktum att det fanns ett stort antal program på marknaden med samma användningsområde, pekade på att programmen inte var resultatet av tekniskt eller logiskt givna premisser. Detta konstaterade domstolen trots att programmen inte

[55] Wagle/ Ödegaard, s173ff.
[56] NJA 1996 s79.

presenterats i sin funktion för domstolen. Domstolen tillade något kryptiskt att det endast är program som uppfyller kravet på verkshöjd som är av intresse för den tilltalade. Bedömningsgrunden att programmen inte var av trivial art p.g.a. mängden instruktioner är med bakgrund till EG- direktivet något förvånande. HD gjorde ingen annan bedömning än hovrätten avseende verkshöjden.

6.4.2 Hovrätten för västra Sverige 19/11 1987.

Det andra fallet jag tänkte nämna är hovrättsmålet från västra Sverige som har berörts ovan, återgivet i NIR 1988 s310. Målet är intressant ur den aspekten att professor Peter Seipel kallades som sakkunnig i målet. Målet rör piratkopiering av datorprogram. Seipel ställer upp tre situationer när datorprogram inte får anses åtnjuta upphovsrättsligt skydd[57], nämligen när

a) programmeringen är trivial,
b) programmeringen sker mekaniskt eller
c) programmet innebär en optimal eller nära nog optimal med hänsyn till förutsättningarna.

Att programmeringen är trivial menas, att utrymmet för den personliga prägeln är obefintlig. Nivån på programmeringen är således så låg att verket inte kan skyddas. Den mekaniska aspekten hör till viss del samman med den triviala. Programmet är av den standarden att det snabbt kan fogas ihop av redan existerande kända moduler. Seipel jämför detta med en kort beskrivning av hur en shunt för fördelning av varmvatten kan utformas. De yttre förutsättningarna sätter gräns för det personliga spelrummet. Det sista kriteriet behandlades i viss omfattning ovan. Även om det råder delade meningar om existensen av optimala program, är definitionen av ett sådant att lösningen på programmet, med hänsyn till förutsättningarna, i så stor utsträckning är styrda av logiska eller tekniska förutsättningar att det inte heller här finns utrymme för personlighet. Hjälpnormen om risken för dubbelskapande har utvecklats i detta avseende. Hjälpnormen är däremot inte invändningsfri. Andersen menar att om man hårdrar normens praktiska tillämpning, finns det risk för att felaktiga program gynnas.[58] Individualiteten och personligheten presenteras i inkonsekventa, felaktiga program, som egentligen skulle falla på dubbelframställningskriteriet men som trots detta speglar skaparens personlighet. Samtidigt finns det en risk för att program som är resultatet av god programmering, drabbas av en hårdare bedömning för att de anses vara för opersonliga. Individualiteten skall istället belysas i det faktum att verket har en sådan komplexitet, vilken möjliggör personlig utformning och i jämförelse med andra program med liknande funktioner trots detta skiljer sig i programmeringen. Hovrätten menade att samtliga program, till skillnad från tingsrätten som ansåg att ett kalkylprogram inte var tillräckligt originellt, åtnjöt upphovsrättsligt skydd,

[57] NIR 1988 s316f.
[58] Andersen Mads Bryde, Laerobog i EDB- ret, s204.

dvs att programmen hade verkshöjd. Gällande kalkylprogrammet menade domstolen att kalkylprogrammen fanns i så många olika varianter att det kunde uteslutas att optimeringsmöjlighet förelåg.

6.4.3 Tippe-domen.

En dansk dom som är intressant i sammanhanget är den s k Tippesystemsaken.[59] Målet gällde ett datorprogram som hade utvecklats för att konstruera tipssystem med en utskriftsfunktion för tipskupongerna, detta för att stora system är ytterst besvärliga att skriva ut för hand. Tipskupongerna är som bekant utformade med 13 stycken spelbara matcher med tre olika alternativa resultatmöjligheter. Systemutvecklaren upptäckte att en konkurrent hade plagierat fem av kärandens tipssystem. Tipssystemen grundade sig på 246 olika kombinationsmöjligheter av grundkoden. Domstolen hade att ta ställning till om programmen hade verkshöjd. Domstolen menade att de grundläggande idéer och metoder är så enkla, att två konstruktörer med stor sannolikhet oavhängigt av varandra skulle komma fram till samma system. Domstolen lade således till grund för sin dom att programmen inte var tillräckligt originella, utan att ett skydd skulle närma sig ett idéskydd i alltför stor utsträckning. Vidare skulle ett skydd begränsa andra programmerares tillgång till området, dvs i det närmaste likna en monopolställning.

Sammanfattningsvis kan man alltså konstatera att det är originalitetskravet som skall stå i centrum för bedömningen om ett program har verkshöjd eller inte. I avvaktan på praxis från EG- domstolen kan endast spekulationer göras i vad som mer konkret ligger i begreppet, där det mest intressanta är huruvida kvaliteten på programmet har någon inverkan på skyddet eller skyddsomfånget.

[59] UfR 1993 s17.

7. Plagiatvärderingen.

7.1 Inledning.

I alla intrångsprocesser inom upphovsrätten, måste käranden bevisa att motparten genom sitt alster har efterbildat kärandens verk.[60] Upphovsrättsligt skydd omfattar inte enbart ren kopiering, utan även, enl 4§ 1 st UrL, översättningar och bearbetningar. Det är i praktiken omöjligt att ställa upp några konkreta riktlinjer för domstolarna att följa vid likhetsbedömningen, utan denna får ske i varje enskilt fall, ofta med experthjälp. Denna bygger ofta på en helhetsbedömning av verket med avseende på en om det finns en *identitetsupplevelse* verken emellan.[61] Det finns inga fall i Sverige från högsta instans som avser likhetsbedömningen av datorprogram. Detta beror till stor del på att tvisterna avgörs i skiljedomstol som sällan offentliggörs. Jag kommer nedan att redogöra för plagiatvärderingen av datorprogram utifrån de allmänna principer som finns inom svensk rätt. Då datorprogram utgör en i många fall säregen del av den litterära verkskategorin, kommer jag även att redogöra för de principer som har utvecklats utomlands, främst i USA. Vid denna jämförelse är det mycket viktigt att man har i åtanke att det amerikanska systemet bygger på en copyrightprincip medan det svenska systemet bygger på principen att skydda andliga verk. USA saknar dessutom en federal marknadsföringslag, vilket sätter press på domstolarna att ge ett bredare skydd för datorprogram.

7.1.1 Krav på kännedom om originalverket.

I intrångsprocesser rörande upphovsrätt, måste det först konstateras att inträngaren haft kännedom om det påstådda kränkta verket.[62] Än viktigare är att det även måste fastställas att inträngaren har utnyttjat originalverket. Detta medför i praktiken stora bevissvårigheter för kärandeparten. Då det är originalskaparens uppgift att bevisa sådan kännedom och att utnyttjande har skett, är det vid de traditionella verkstyperna, såsom litterära verk och brukskonsten, vanligt att originalskaparen försöker fastställa detta genom att visa att inträngaren genom exempelvis marknadsföring av originalprodukten inte kunnat undgå att låta sig påverkas av originalverket vid skapandet av det egna verket, jfr exempelvis NJA 1995 s164 Stickad Tunika. Rörande datorprogram är situationen ofta annorlunda, framför allt de situationer då plagiat eller bearbetning är för handen. När datorprogram säljs levereras de i de allra flesta fall inte i källkodsversion utan i objektkod. Vid plagiatvärderingen av datorprogrammets kod, är det just källkoden som får ses som utgångspunkten vid jämförelsen. Att jämföra programmens respektive objektkoder är ett i praktiken omöjligt arbete att genomföra. I de

[60] NJA 1994 s74
[61] Koktvedgaard Mogens, Levin Marianne, s139.
[62] Koktvedgaard Mogens, Levin Marianne, s146.

fall som jag nedan kommer att redogöra för, rör det ofta sig om situationer när en programutvecklare skapat ett program för ett företag i en relativt specifik bransch och därefter utvecklat ett nytt program till ett konkurrerande företag. Svårigheterna med att bevisa vetskap respektive utnyttjande av originalprogrammet är därför i de fall betydligt lättare.

7.1.2 Kod kontra genererad ut- effekt.

Inom programvaruindustrin är det oerhört viktigt att programmet är användarvänligt, och detta gäller givetvis i första hand bruksprogrammen. Känner användaren sig hemma i programmets ”miljö” i form av skärmbild eller funktionstangenter, är det troligt att han vid senare programanskaffande väljer ett program från samma leverantör. Som exempel kan ges att Microsoft medvetet satsar stora resurser på att göra användaren van vid programmiljön för att på så sätt knyta till sig kunder vid senare programinköp. När Netscape introducerade sin webbläsare Navigator på marknaden, skapade de som första producent en användarmiljö inom Internettekniken som slog igenom kommersiellt. När Microsoft förstod marknadsvärdet i produkten, skapade de på mycket kort tid den konkurrerande webbläsaren Explorer. Det krävs endast en hastig jämförelse mellan de båda webbläsarna, i varje fall de tidigare versionerna, för att konstatera att Explorers gränssnitt mer eller mindre är en klon av Netscape. En direkt jämförelse mellan gränssnittet och övriga användarmiljön innebär dock inte att det behöver föreligga plagiat mellan programkoderna. Detta gäller givetvis även vice versa. Plagiat av programkoden kan med enkla ändringar i kodens struktur som reglerar användargränssnittet generera vitt skilda skärmutseende och funktionstangenter. Detta innebär att datorprogrammen intar en särställning inom upphovsrätten i intrångsprocesserna. Själva koden är uttryckligen skyddad av lagen men det stora problemet är att plagieringen av programkoden inte nödvändigtvis resulterar i synbara effekter som skvallrar om en eventuell plagiering. När jag nedan kommer att redogöra för intrångsprocesserna är det viktigt att skilja på plagiering av själva programkoden och likhetsbedömningar i de effekter som programmet genererar. I de fall jag skall redogöra för kommer det dock att visa sig att det i många fall inte finns några vattentäta skott mellan de båda situationerna.

7.2 Idé kontra utformning.

I samtliga länder till vilka jag kommer att hänvisa rättsfall, är det en väl grundad uppfattning att det är verkets konkreta form som skyddas, inte idén, verkets motiv, metod eller grundläggande principer. Detta finns implicit uttryckt i UrL 4§ 2st, att verk som har tillkommit i fri anslutning till ett verk, inte är beroende av rätten till originalverket. Även i EG-direktivet för skydd för datorprogram är detta uttryckt i art 1.2. Orsaken till detta är självklar; ett idé- eller principskydd skulle innebära stor risk för monopolställning och skulle resultera i stora hinder i utvecklingen av datorprogram. Idén att skapa ett ritprogram skulle tillhöra en upphovsman och programmeringstekniker i

samma utsträckning. Det är i praktiken dock väldigt svårt att dra en konkret gräns mellan idé och utformning, utan det är en bedömning som får göras i varje enskilt fall. I svensk rättspraxis kan nämnas ett äldre rättsfall, NJA 1938 s497. Fallet rör ett påstått upphovsrättsligt intrång i en film baserad på samma manuskript. Domstolen kom fram till att verkets grundidé var densamma, det ursprungliga manuskriptet och att handlingen i de båda verken byggde på denna grundidé. Fallet är egentligen inte så representativt för att utröna var idén tar slut och utformningen av verket börjar. Båda manuskripten visade sig bygga på teaterpjäsen Pygmalion och utformningen av de båda alstren var följaktligen en direkt förlängning av samma grundidé, pjäsens manuskript. De båda utformningarna resulterade således helt naturligt i två filmverk som i utformningen hade stora likheter. En bedömning av verkens förhållande till det ursprungliga manuskriptet Pygmalion gjordes givetvis inte.

7.2.1 Sybordsdomen

Det finns ett intressantare exempel från den norska högsta domstolen som kan komma till användning för bedömningen av datorprogram. Fallet är från 1962 och rör en möbel som användes som ett kombinerat sy- och soffbord, Rt. 1962 s964. Möbeln tillverkades av en dansk möbelfabrik och var designad av en känd möbelkonstruktör. Det framkom sedermera att en norsk möbelfabrik konstruerat en motsvarande produkt men till ett avsevärt lägre pris. Den danska fabriken och konstruktören menade att det norska bordet utgjorde intrång i ensamrätten att framställa exemplar av bordet. Svaranden ansåg dels att bordet inte var resultatet av andligt skapande och därför inte skyddades av lagen om upphovsrätt, dels att om det skulle visa sig att bordet åtnjöt upphovsrättsligt skydd utgjorde svarandes part inget intrång i kärandes bord. Jag ämnar koncentrera mig främst på likhetsbedömningen, då domstolen kom fram till att bordet var resultatet av ett andligt skapande och att skydd kunde åtnjutas. Beträffande möbelindustrin är det speciellt känsligt att gränsen mellan idé och utformning upprätthålls på en rimlig nivå. Skyddsomfånget får inte bli för vitt, då risken för monopolställning i framtiden kan hindra design av elementära objekt som möbler. Situationen kan jämföras med datorprogram. Ett alltför vitt skydd för delrutiner i ett datorprogram kan även det lägga hinder i vägen för programvaruutvecklingen. Bordet, som jag har förstått det, hade en form av sykorg fäst vid bordets kortända. Domstolen konstaterade på ett tidigt stadium att själva idén med ett kombinerat soff- och sybord inte kunde åtnjuta skydd. Placeringen av korgen samt materialvalet var dock identiska på de båda borden och det var detta som domstolen hade att ta ställning till nämligen om en identisk placering utgjorde en kränkning i kärandens upphovsrättsliga skydd. Domstolen ansåg att det inte fanns någon anledning att göra någon skillnad på idén att skapa ett kombinerat sy- och soffbord med materialvalet och korgens placering. Detta med avseende på att dessa faktorer styrdes av praktiska hänsyn som var en direkt förlängning av idén. Valmöjligheterna för utformningen var ytterst begränsade och den mest naturliga placeringen var just den de fått i de båda alstren. Det var alltså rent

praktiska faktorer som styrde placeringen och materialvalet och dessa var en direkt och nödvändig förlängning av idén. Med andra ord satte domstolen likhetstecken mellan utformningen av sybordet och idén med densamma.[63]

7.2.1.1 Kommentarer

Slutsatsen man kan dra av detta fall är således att det verkar vara så att skyddsomfånget är direkt relaterat till ju fler alternativa lösningar som upphovsmannen har att utforma sitt verk. Motsvarande resonemang har förts inom mönsterrätten i exempelvis NJA 1990 s168, Saxverktyg. Det norska fallet väcker vissa likheter med det ovan refererade hovrättsfallet, HovR f. Västra Sverige 19/11 1987[64], där frågan prövades om ett datorprogram överhuvudtaget kan åtnjuta upphovsrättsligt skydd. Professor Seipel konstaterade att så var fallet och hade som en förutsättning för skydd att den intellektuella insatsen inte var en direkt förlängning av en antagen optimal programmeringsteknik. De reella valmöjligheterna samt även yttre begränsningar i form av programmets möjligheter till olika utformningar bestämmer sålunda dels om programmet når upp till verkshöjdsgränsen och åtnjuter upphovsrättsligt skydd, dels skyddsomfångets storlek. Har verket väl nått upp till skyddsbar nivå, är det istället de yttre faktorerna som bestämmer skyddsomfånget. Likhetsbedömningen skall ju bygga på en identitetsupplevelse verken emellan. Detta måste skiljas från en helhetsbedömning av verken. HD har i det sk Smultronfallet rörande brukskonsten, uttalat att trots att helhetsintrycken mellan två tygmönster är lika, gör den påstådda inträngaren inte intrång i originalverkets upphovsrätt. ”Sammantaget är dock olikheten så stor, att jordgubbsmönstret inte kan sägas ligga inom skyddsomfånget för ’Smultron’ ”.[65] Helhetsbedömningen är direkt avhängig skyddsomfånget och således finns det paralleller mellan det svenska och det norska fallet. Skyddsomfångets vidd är beroende av antalet tänkbara alternativa utformningsmöjligheter för verket kombinerat med en identitetsupplevelse samt en helhetsbedömning. Den mest påfallande slutsatsen är dock att några generella regler för när idén tenderar att övergå till en skyddad utformning inte är möjlig att göra, utan får konstateras i varje enskilt fall.

7.3 Strukturell uppdelning

Ensamrätten till exemplarframställning i UrL 2§, innebär ofta inga problem med värderingen om intrång i ensamrätten har gjorts. Vid en direkt jämförelse mellan källkoderna kan denna jämförelse ske relativt smärtfritt. Så var fallet i det ovan nämnda hovrättsavgörandet angående piratkopiering av spel- respektive kalkylprogram. Det upphovsrättsliga intrånget kunde konstateras lätt, då det rörde sig om ren kopiering. Det som ger upphov till svårare bedömningar och problem, är när gränsen till bearbetning i UrL 4§ 1 st, respektive nytt verk i fri anslutning till originalverket, UrL 4§ 2 st, skall konstateras. Stensaasen kategoriserar, enligt nedan, de olika situationerna

[63] Wagle Mediaas Anders, Ödegaard jr Magnus, s345.
[64] NIR 1988 s317.
[65] NJA 1994 s74.

som kan uppkomma, vilka jag kommer att hålla mig till för att få en mer överskådlig struktur:[66]

1. Kopiering.
2. Plagiat.
3. Bearbetning.
4. Nytt verk i fri anslutning till originalverket.

Gränsdragningen mellan plagiat och bearbetning är viktig men aldrig lätt att konstatera, utan även den bedömningen överlämnas till domstol eller skiljenämnd vid eventuell tvist. Gränsdragningen är viktig i det avseendet att den som har skapat en bearbetning äger upphovsrättsligt skydd för denna men har inte rätt att fritt råda över verket, utan denna rätt är direkt avhängig av rätten till originalet. Detta innebär att upphovsmannen till bearbetningen t ex inte får offentliggöra verket utan samtycke av upphovsmannen till originalverket.[67] Det som kategoriserar en bearbetning är att det finns ett moment av intellektuellt skapande insats av upphovsmannen. Denna insats är dock inte så påtaglig att identiteten hos verket är ändrat så att bearbetaren kan råda över verket på egen hand.

Plagiatet befinner sig mellan kopian och bearbetningen. Skaparen har inte bidragit med sådan intellektuell instats att verket kan anses vara originellt i den omfattningen att en bearbetning av originalverket har skett. Samtidigt har inte en direkt kopiering av källkoden skett som skulle klassificera verket som en kopia. Plagiatet befinner sig inom skyddsomfånget för originalverket och ger skaparen till originalverket ensamrätt till det. Plagieraren har överhuvudtaget inte någon upphovsrätt till skapelsen till skillnad från den som har skapat en bearbetning.
Svårigheterna att fastställa om det har tillkommit ett nytt verk i fri anslutning till originalverket, brukar hänföras till begreppen yttre och inre form.[68] Den inre formen är den skyddade och med det menas hur en abstrakt idé har kommit till uttryck, medan den yttre formen hänför sig till i vilken form uttrycket är materialiserat, t ex en bok eller en tavla.[69] Till skillnad från andra verkskategorier, är denna jämförelse i många fall enklare att göra vid datorprogram, såvida källkoderna finns till hands.

Det uppstår redan på detta stadium rena praktiska problem som sätter datorprogram i en särskild position jämfört med både litterära verk och andra verkskategorier. Som jag nämnde ovan kan även små ändringar i programkoden, detta gäller även mycket omfattande program, resultera i att ut- funktionerna i form av användargränssnitt, ljudeffekter och skärmbild, är totalt skilda från originalprogrammet. De praktiska problemen resulterar

[66] Stensaasen Tarjei, s62.
[67] Koktvedgaard Mogens, Levin Marianne, s139.
[68] SOU 1956:65 s68f.
[69] Nordell Per Jonas, Om skyddsomfång vid upphovsrättsligt skapande, särskilt med hänsyn till skapande med hjälp av datorteknik, NIR 1991 s374.

således i att det ofta kan vara mycket svårt att upptäcka intrång i originalskaparens program.
Det teoretiska problemet som måste fastslås är när det föreligger en tillräckligt originell intellektuell ändring av verket för att det skall kunna klassificeras som bearbetning av originalverket. Precis som vid gränsdragningen för verkshöjden för alla typer av verk är denna gräns ytterst svår att precisera. Jag hade nedan tänkt att, i vissa fall med exempel från praxis, försöka konkretisera denna gräns. Den slutgiltiga bedömningen kommer givetvis att ske i domstol, där resultatet är avhängigt varje enskilt fall.

7.3.1 Ändring av variabelnamn.

Ett datorprogram som når upp till den originella intellektuella insats som krävs för att verket har verkshöjd innehåller ofta flera hundra, i många fall ännu fler, variabelnamn. Variabelnamnen som sådana har ingen styrande funktion för programmets yttre eller inre funktioner, utan de benämns vid namn för av rena praktiska skäl att man skall kunna relatera namnet till en viss kategori som programmets ut- funktion hänvisar till. Som ett enkelt exempel kan nämnas ett medlemsregister i form av ett program med uppdateringsmöjligheter. Tänkbara variabler i ett dylikt program är medlemskategori, namn, adress, etc.. Vid skapandet av programmet är det givetvis enklast att döpa de variablerna som styr namninmatningen för ”namn”. Detta för att underlätta användarvänligheten för den som skall bruka programmet. Många program, t ex Pascal, arbetar dessutom i procedurformer. Även här döper man procedurer, t ex utskriftsproceduren till ”utskrift”, för att i ett senare skede i programmet kunna kalla på nämnda procedur utan att behöva skriva om densamma ytterligare en gång. Variablerna som sådana har inget med själva programlogiken att göra, dvs styr inga logiska processer i programmet. En tänkbar situation är att en person kommer över källkoden till programmet, för att sedan ändra variablerna efter det syfte som passar honom. Istället för ett medlemsregister vill personen skapa ett kundregister för sin verksamhet. De variabler som kallas medlemskategori döps om på samtliga ställen till ”kund”. Programmets logiska uppbyggnad är därmed identisk med originalprogrammet men skillnaden består i ut- funktionerna på skärmen, vilket visar sig om en strukturkarta görs över de båda programmen. Denna typ av ändringar kan inte anses som tillräckligt kreativa i den bemärkelsen att en intellektuell skapelse har bidragit till verket så att en bearbetning av verket har skett. Ändringarna ligger inom skyddsomfånget för originalverket och det är fortfarande endast upphovsmannen till originalverket som får förfoga över det.

7.3.2 Hjälptexter mm.

I ett datorprogram finns det ofta utrymme att lägga in data som är skrivet i naturligt språk, bl a hjälptexter. Dessa hör inte till programmet som sådant, utan skyddas under de traditionella litterära verken. Dessa data påverkar således inte programmets logiska struktur. En ny utformning av dessa

medför därför inte att någon ny insats har skett vad beträffar själva programmet. Precis som i exemplet med variabelnamnen kommer strukturkartorna på de båda programmen att vara identiska. Det är dock inte otänkbart att de nya data som har lagts in resulterar i ett nytt litterärt verk och därmed ger upphovsmannen skydd för sitt litterära verk. Själva förfoganderätten över datorprogrammet som verk har dock originalskaparen fortfarande ensamrätt på.

7.3.3 Delar infogade.

I förarbetena till införandet av datorprogram i UrL, togs frågan upp huruvida skydd som samlingsverk enligt 5§ UrL kunde åtnjutas för datorprogram sammanfogade av moduler av redan existerande program.[70] Sådana program kan åtnjuta skydd såvida modulerna som programmet är sammanfogat av uppfyller verkshöjdskravet och att verket som sådant uppfyller verkshöjdskravet. Är verkshöjdskravet för modulerna inte uppfyllt, kan verket åtnjuta skydd som originalprogram, såvida verkshöjdskravet för samlingsverk är uppfyllt. Problemet är att ett datorprogram sammansatt av moduler inte är ett samlingsverk av klassisk natur, såsom en antologi eller en diktsamling. En diktsamling har just funktionen av att sammanställa olika dikter till ett verk medan ett datorprogram sammanställt av moduler inte har karaktären av att sammanställa något. En jämförelse kan göras med en film som trots att den består av skilda verk inte betraktas som ett samlingsverk enl 5§ UrL. Modulernas sammanställning bidrar istället till att skapa ett nytt datorprogram. Man kan säga att modulerna spelar en aktiv funktion i programmet, till skillnad från delverken i antologin som har en mer passiv funktion[71]. Har dock verket en sammanställningsfunktion kan det åtnjuta skydd som samlingsverk, med förfoganderätt som är beroende av de enskilda verken.
Bedömningsproblem om intrång har skett kan dock uppstå när moduler, som var för sig uppfyller verkshöjdskravet, infogas i ett program. Skall man betrakta programmet utifrån en helhetssyn eller de enskilda delarna var för sig? Med resonemanget ovan med inre och yttre form borde frågan inte skapa några problem. Ett nytt och självständigt verk kan ha skapats, även om programmet består av skyddade delar, såvida den inre formen i det nya verket inte är densamma.[72] Två faktorer gör dock bedömningen problematisk. Den första är en kritiserad HD- dom, NJA 1990 s499.[73] HD verkade bortse ifrån att den inre formen kan vara ändrad även om verket består av var för sig skyddade verk. Den andra faktorn är hur man bedömer att den inre formen har ändrats i ett datorprogram och vad som omfattas av den inre formen. Ett datorprograms uppbyggnad är ofta så omfattande, att det är svårt att utkristallisera en inre form. Den svenska helhetssynen är därför svårtillämpad på datorprogram, särskilt i det avseendet då

[70] SOU 1985:51 s92.
[71] Nordell Per Jonas, NIR 1991 s378f.
[72] Nordell Per Jonas, NIR 1991 s379.
[73] Målet rörde en plastkasse med motiv av en gotlandkarta. Motivet var uppbyggt av nio mindre teckningar.

bedömningen sker av de icke- litterära elementen i ett program, se nedan. Att applicera ett helhetsperspektiv på verket är inte heller överensstämmande med den anglosaxiska plagiatvärderingen. I England utgår man inte från att verk kan bearbetas, utan har istället ett väsentlighetskriterium i bedömningen om plagiering har skett, vilket skiljer sig från den svenska helhetsbedömningen vid bearbetningar. [74] Det finns tecken som tyder på att den anglosaxiska traditionen har påverkat utformningen av EG- direktivet och därmed direkt påverkar svenska förhållanden. En möjlighet är därför att helhetssynen på en bearbetning i viss mån skall överges vid bedömning av datorprogram. Det skulle innebära att även om den inre formen i de båda verken inte är densamma, kan intrång ändock föreligga om något av de väsentliga inslagen i ett program även finns i den påstådda bearbetningen.[75] Frågan är inte prövad i praxis så något konkret svar kan inte ges.

7.3.4 Olika programspråk.

I 4§ 1 st UrL, står det uttryckligen att översättaren till ett verk äger upphovsrätt till det översatta verket men äger ingen förfoganderätt över originalversionen. En översättning jämställs med en bearbetning och ställer inte till med några bevisproblem för de traditionella litterära verken, då det är lätt att kontrollera om en översättning av ett litterärt verk har skett. För datorprogram är situationen inte lika lättöverskådlig. Datorprogram översätts ofta då vissa maskiner kräver vissa standarder som inte ”förstår” alla språk. I förarbetena nämns situationen hur en av översättning av programspråk med ett översättningsprogram skall bedömas.[76] Man skiljer här på två situationer. Den första är när översättning sker helt automatiskt med hjälp av ett program. Någon intellektuell instats sker inte, utan programmet blir endast representerat på ett annat sätt, med konsekvensen att ”översättarens” version inte blir upphovsrättsligt skyddad. Det som inträffar är att när programmet översätts sker ingen egentlig förändring i själva programlogiken. Algoritmen kommer att vara oförändrad men den kommer att uttryckas i ett annat språk. Jämför man de olika programmens strukturkartor kommer de till stor del att vara lika utformade, programmens identitet kommer i stort sett att vara den samma. Vid de tillfällen då översättningen inte kan ske helt automatiskt, utan en intellektuell insats krävs, finns dock utrymme för ”översättaren” att åtnjuta upphovsrätt till den översatta versionen. I praxis finns det ett norskt underrättsfall som behandlar konvertering av ett program.[77] Domen är från mitten av 1980- talet och det är svårt att lägga någon större vikt vid resultatet. Det mest intressanta med domen är egentligen att det var ett av de första fallen i Norge där frågan om datorprogram kunde åtnjuta upphovsrättsligt skydd. Fallet rörde ett program som hade konverterats i stor omfattning. Domstolen kom fram till att konverteringen var en bearbetning

[74] The Copyright, Design and Patents Act, Art 16 (3).
[75] Plogell Michael, s58f.
[76] SOU 1985:51 s92.
[77] Oslo byrett 18/3 1986.

av ursprungsprogrammet. Den omedelbara slutsatsen man får dra är dock att varje enskilt fall får bedömas var för sig.

7.3.4.1 Ickebesläktade språk

Förarbetena tar dock inte upp situationen när ett översättningsprogram inte kan användas. Den första situationen är den när språken inte är så nära besläktade med varandra, dvs den systematiska uppbyggnaden skiljer språken i så stor utsträckning att någon automatisk konvertering inte kan göras med hjälp av ett konverteringsprogram. Det skall tilläggas att när språkskillnaderna är så pass stora, är det ofta inte ekonomiskt lönsamt att göra en konvertering, utan det är både mer tids- och kostnadseffektivt att utveckla ett nytt datorprogram. I denna situation är det inte helt säkert att frågan som skall ställas är om en bearbetning har skett, utan istället var gränsen går mellan bearbetning av programmet och om ett nytt verk har skapats i fri anslutning till originalverket enligt 4§ 1 st UrL.[78]
Konverteringen kräver så stora ansträngningar och en närmast total omarbetning av programmet, att det ofta endast är den bakomliggande idén till programmet som finns kvar. Detta visar sig även om programmens strukturkartor jämförs. Systematiken i programmen är så olika att det vid en jämförelse inte kommer att finnas någon identitetsupplevelse kartorna emellan, dvs verkens inre form är inte längre densamma. Denna bedömning måste likväl göras i varje enskilt fall och resultatet är beroende på hur nära släktskap språken har. Situationen kompliceras dessutom av att en bedömning får göras av verkets icke- litterära element, se nedan.

Den andra situationen är när det föreligger generationsmässiga skillnader i språken. Situationen motsvarar den när språkskillnaderna är stora, med det tillägget att en ännu mer omfattande omarbetning av programmet måste ske. Situationen torde därför i samtliga fall vara den att ett nytt verk har skapats, dvs att den inre formen är annorlunda i de båda programmen. Liknande situation uppkommer vid reverse engineering, där problemet inte är utrett i praxis. I UrL 26h§ 2 st 3p uttalar att den information som fås genom operationen inte får användas till utvecklande av program, vilka har en väsentligt likartad uttrycksform. Jan Rosén menar att om den kompilerade versionen ligger till grund för ett program skrivet i ett annat programspråk, implicit under förutsättning att det krävs en kreativ insats av en programmerare, hindras denne inte av förbudet i UrL 26h §.[79]

7.3.4.2 Q- Co Industries

Ett intressant amerikanskt rättsfall som rör just översättning av programspråk med svagt släktskap är Q- Co industries, Inc. v. Hoffman.[80]
En tidigare anställd programutvecklare på Q- Co hade översatt ett program ägnad åt en Atari- dator till en IBM- dator. Avstånden mellan språken för dessa maskiner är avsevärt stora och domstolen kom således fram till att

[78] Schmidt Per Håkon, s233.
[79] Rosén Jan, s39f. För avvikande uppfattning se Plogell Michael, s55.
[80] Q- Co. Industries, Inc. v. Hoffman, 625 F.Supp. 608 (S. D. N. Y. 1985).

något plagiat inte förelåg. Själva processen att översätta programmet krävde en sådan total omarbetning att det som kvarstod var endast den överordnade idén för programmet. Det är inte otänkbart att fallet, ur en upphovsrättslig synvinkel, hade bedömts lika i Sverige. Det är dock inte otänkbart att den anställde istället hade dömts för brott mot lagen om företagshemligheter. Den amerikanska domstolen gjorde en liknade bedömning.

7.4 Skyddet för icke- litterära element.

”If it Looks Like a Duck, Walks Like a Duck, and Quacks, it Must be a Duck.”[81]

När verk har skrivits om i ett annat programspråk, uppkommer således frågan om ett nytt verk har uppkommit, en bearbetning har skapats eller om en otillåten exemplarframställning har skett. Den egentliga frågan som skall besvaras grundas i botten på var gränsen mellan idé och utformning går men även om skyddet för datorprogram sträcker sig bortom en direkt jämförelse mellan programkoderna. Frågan är om även de icke- litterära elementen i ett program skyddas, såsom strukturen i programmet men även det som programkoden genererar. Då det saknas praxis i Norden har blickarna vänts mot USA, där intrångsprocesser rörande datorprogram i en helt annan utsträckning förekommit ända sedan datorprogrammen gjorde sitt intåg på den upphovsrättsliga arenan. Jag hade därför tänkt redogöra för ett antal internationellt mycket uppmärksammade rättsfall för att belysa hur man har tacklat problemet i USA. Därefter undersöker jag om de amerikanska rättsfallen även har någon relevans för svenska och europeiska förhållanden.

Datorprogram är sedan 1980 uttryckligen skyddade i Copyright Act, 17 U. S. C. § 102. På basis av ett antal tidiga avgöranden är det fastställt att både objekt- och källkoden är skyddade.[82] Precis som i Sverige kan inte idéer vara skyddade vilket uttryckligen finns i Copyright Act § 102 (b), vilken lyder:

> ”In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle or discovery, regardless of the form in which it is described, explained, illustrated or embodied in such work.”

[81] Coleman Susan E, The State of Software Copyright Protection in the United States, NIR 1988 s122.
[82] Det mest kända och oftast refererade är Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 219 U.S.P.Q.113 (3d Cir. 1983).

Principen har dock varit fast rotad i det amerikanska rättssystemet efter ett gammalt och ofta refererat rättsfall från högsta domstolen, som för första gången fastslog principen; Baker v. Selden.[83] Baker hade givit ut en handbok om bokföring som byggde på allmänna principer inom ämnet och hade utformat dessa efter egna komponerade tabeller och scheman. Högsta domstolen menade att dessa principer inte var skyddade och därmed utgjorde boken inte ett intrång i copyrightskyddet.

7.4.1 Whelan v. Jaslow.

Upphovet till många av de rättsfall som uppkom under datorprogrammens expansion grundar sig i kontroversen mellan den skyddade programkoden och datorprogrammens funktionsbaserade karaktär. Idén att skapa ett kalkylprogram kan aldrig vara skyddad, kontra datorprogrammens natur att, med avseende på de vitt skilda programspråkens struktur, kunna få identiska funktioner och användargränssnitt. Vad de amerikanska domstolarna hade att ta ställning till var således om icke- litterära element i ett program var skyddade och i vilken utsträckning det krävs väsentlig likhet för att intrång skall kunna konstateras, den så kallade ”look and feel- doktrinen.[84] Det första, och mycket uppmärksammade, fallet på delstatsnivå var målet Whelan v. Jaslow.[85]

7.4.1.1 Bakgrund.[86]

Jaslow, som drev en tandteknikerverksamhet, hade uppdragit åt Whelan att utveckla ett administrationsprogram för att underlätta faktureringen för verksamheter med liknande inriktning. Ett samarbete parterna emellan utvecklades och Jaslow skulle verka som marknadsförare av programmet mot att Whelan fick del av försäljningsintäkterna. Själva utvecklandet av programmet krävde långtgående insatser från Whelan, som först var tvungen att sätt sig in verksamhetens uppbyggnad för att kunna skapa ett så skräddarsytt program som möjligt. Själva kodningen av programmet svarade mot en mindre del av hela utvecklingen av programmet och strukturingen stod för en större del. Programmet skrevs i ett språket EDL som var anpassat för en tidig IBM modell. Jaslow insåg att det fanns en betydande marknad för slika program och utarbetade ett nytt program anpassat för en mer utbredd modell av IBM- datorerna och sade därefter upp kontraktet med Whelan. Programspråket som Jaslow använde i sitt nya program var BASIC. Då språken var olika var det inte frågan om en direkt kopiering av koden, och avstånden mellan språken är så stora att det inte rörde sig om en direkt översättning. Jaslow hade helt enkelt tagit ursprung i Whelans källkod och skrivit ett motsvarande språk kompatibelt med de mer utbredda IBM-datorerna och trots detta hävdade Whelan att Jaslow hade gjort intrång i upphovsrätten för originalprogrammet och syftade på programmets icke-

[83] Baker v. Selden, 101 U. S. 99 (1879).
[84] Byer, David J, Gall, Nicholas L, http://www.tht.com/204875.htm 990217.
[85] Whelan Associates v. Jaslow Dental Laboratory Inc., 797 F. 2d, 1222 (3d Cir. 1986).
[86] Whelan v. Jaslow, s1224f.

litterära aspekter såsom ”Structure- Sequence and Organization”, (SSO), dvs programmets struktur, organisering av moduler etc.

7.4.1.2 Målets utgång.
Rätten ställdes alltså inför den föga lättlösta uppgiften att definiera var programmets skyddade utformning tog vid den oskyddade idén. Med hänvisning till Baker v. Selden- fallet konstaterade domstolen att utformningen skiljs från idén genom att

> ”… the purpose or function of a utilitaitan work would be the work´s idea, and everything that is not necessary to that *purpose or function* would be part of the expression of the idea”.[87]

Rätten konstaterade således att det som inte var en nödvändig förlängning av programmets idé, åtnjöt skydd.[88] Efter en jämförelse mellan strukturen i de båda programmen kunde därför det skyddsvärda området för datorprogram utsträckas till faktorer bortom en direkt jämförelse mellan själva programkoderna. Det som är kontroversiellt med Whelan v. Jaslow- domen hänför sig till två aspekter:

- Det första är att genom att konstatera att den skyddade utformningen av programmet distingeras från idén med det ovan nämnda nödvändighetstestet, resulterar det i att från varje program, som helhet, kan endast härledas en idé. Detta medförde i Whelan- fallet att idén i originalprogrammet är att effektivisera driften av en tandteknikerverksamhet.

- Det andra och mest kontroversiella ståndpunkten är, att vid jämförelsen mellan de båda programmen skall denna ske utifrån ett totalt perspektiv av programmen, inte endast till den uttryckligen skyddade programkoden utan även till programmets struktur och organisation utifrån en generell jämförelse mellan programmen. Detta uttrycks som ”… we are concerned with the *overall* similarities between the programs …”.[89] Denna generella jämförelse omfattar inte bara strukturen i koden, utan även en jämförelse i de funktionella och det visuella användargränssnittet, som programkoden genererar.

7.4.1.3 Kommentarer
Domen kom att skapa stor förvirring främst bland mjukvarutillverkare i landet men kom även att kritiseras hårt av stora delar av den amerikanska doktrinen.[90] Själen till detta är givetvis det stora skyddsomfånget som

[87] Whelan v. Jaslow, s 1236 [min kursivering].
[88] Russo Jack, Nafsiger, Jamie, Software ”Look and feel” protection in the 1990´s. http://www.computerlaw.com/lookfeel.html 990217.
[89] Whelan v. Jaslow, s 1248. [min kursivering].
[90] Se bl a Coleman Susan E, NIR 1988 s123.

domen gav datorprogram men även den osäkerhet domen resulterade i. För det första gav domen upphov till oklarheter hur själva likhetsbedömningen av koderna skulle ske när programmen är skrivna i olika språk. Domstolen hade mer gett generella riktlinjer för att plagiatvärderingen skulle ta avstamp i en generell likhetsbedömning mellan programmen och inte en ingående analys av själva uppbyggnaden av programmet. Med avseende på att distinktionen mellan idé och utformning skedde på ett generellt plan där de icke nödvändiga elementen i ett program kategoriseras som programmets idé, med resultatet att endast en huvudidé kan härledas ur programmet, gav domen ytterst otydliga instruktioner hur denna värdering skulle gå till. För det andra orsakade domen förvirring avseende användargränssnittets betydelse i jämförelseprocessen. Vid denna tidpunkt inleddes den stora expansionen av hemdatorförsäljningen och användargränssnittet hos ett program började således att betyda oerhört mycket för etableringen av nya produkter. Koden som genererar gränssnittet har liten betydelse i själva kodens omfattning men i själva plagiatvärderingen kom den som resultat av Whelan- domen att betyda en hel del.

Domen blev startskottet för en intensiv diskussion rörande skyddsomfånget för datorprogram. Den följdes i många fall i USA, med vissa avvikelser men får anses vara överspelad idag i och med två senare domar vilka jag nedan kommer att redogöra för. Domen är dock intressant ur ett svenskt perspektiv, då den ger upphov till tre separata intressanta frågeställningar:

1. Hur skiljer man oskyddade moment i koden från skyddade sådana, dels med avseende på fria element, dels med avseende på idé kontra utformning?

2. I vilken omfattning skyddas de icke- litterära momenten i ett program, dvs vilket skydd finns för SSO.

3. Vilken betydelse har användargränssnittet, med bildskärmen och funktionskontroller, för plagiatvärderingen?

7.4.2 Altai-fallet.

Det första fallet som rör den första frågeställningen är Altai- domen.[91] Målet var intressant redan i förväg, eftersom det var första gången 2nd Circuit prövade ett fall som rörde datorprogrammens skydd, då domstolen är känd för sin kompetens i copyrightmål.[92]

[91] Computer Associates International, Inc. v. Altai Inc., 982 F. 2d, 693 (2d Cir. 1992).
[92] Branley Thomas H., Is copyright protection for ”structure, sequence and organization” dead? http://www.lgu.com/cr43.htm 990217.

7.4.2.1 Bakgrund.[93]

Computer Associates (CA) hade utvecklat ett program som de kallade CA-scheduler, ett styrprogram för IBM- datorer, vars uppgift var att specificera och kontrollera instruktioner till datorn. En del av programmet bestod av ett översättningsprogram, ADAPTER, vars syfte var att underlätta driften av programmet för datorer med olika operativsystem. Avsikten med ADAPTER var således att översätta CA- schedulers hänvisningar till operativsystemet oavsett om det var MS- DOS, MVS eller CVM. Den slutgiltiga poängen var att endast en version av CA- scheduler behövdes skrivas, utan att anpassning behövde ske till vilket operativsystem som användes i datorn.

Altai var ett konkurrerade företag som sålde ett liknade program, ZEKE, med den stora skillnaden att det endast var kompatibelt med operativsystemet VSE, vilket innebar en betydande konkurrensnackdel. Altai ville därför utveckla programmet så att det dessutom blev kompatibelt med MVS och därför rekryterades en person med tidigare anställning på CA, Arney. Arney var väldigt insatt i ADAPTER- programmet, vilket rekryteraren på Altai inte var medveten om. Arney hade dessutom tillgång till källkoden av ADAPTER, vilken han helt sonika tog med sig till Altai och använde vid utvecklingen av det nya programmet. Väl medveten om ADATPTERs fördelar, kopierade han 30% av koden och använde den i den första versionen av det nya programmet, OSCAR 3.4, något som ingen annan på Altai var medveten om.

Efter att det nya programmet introducerats på marknaden väcktes misstankar hos CA att delar av ADAPTER hade kopierats och infogats i bl a det nya versionen av ZEKE. När detta kom till kännedom hos Altai omarbetades OSCAR 3.4 till en nya version, 3.5, utan att använda sig av källkoden till ADAPTER, av programmerare som inte hade varit med vid utvecklingen av 3.4 versionen och som inte hade någon kontakt med Arney. CA hävdade dock att även Altais nya version av OSCAR gjorde intrång i ADAPTER, trots att källkoden var skriven i ett annat språk. Frågan som domstolen hade att ta ställning till var om den nya versionen av OSCAR utgjorde intrång i ADAPTER, dvs i vilken utsträckning de icke- litterära momenten i programmet åtnjöt skydd.

7.4.2.2 Fler än en bakomliggande idé

Domstolen inledde med att rikta skarp kritik mot domstolen ställningstagande i Whelan- domen, att endast en idé ligger till grund för ett program.[94] Det ställningstagandet baserar sig på en grundläggande missuppfattning av programmens uppbyggnad. Datorprogram består av moduler och subrutiner som aktivt samverkar för att generera ett resultat. Det beskriver således programmets verklighet felaktigt genom att applicera ett helhetsperspektiv på den bakomliggande idén. Modulernas samverkan

[93] Computer Associates v. Altai, s698f.
[94] Computer Associates v. Altai, s705.

innebär istället att idéperspektivet får appliceras på varje enskild modul och subrutin. Domstolen uttrycker detta genom att

> ”…each subroutine is itself a program, and thus, may be said to have its own ”idea” …”[95]

Detta uttalande innebar ett väsentligt avsteg från de ställningstaganden som gjordes i Whelan- saken.

7.4.2.3 Trestegsprocessen

Den mest intressanta iakttagelsen av Altai- domen är dock den metod domstolen använde för att undersöka om det påstådda programmet visade påtaglig likhet med originalprogrammet. Programspråken för de båda programmen var olika, så någon direkt jämförelse mellan källkoderna kunde inte ske. Testet kallas för Abstraction- Filtration- Comparison- test, men jag använder termen Trestegsprocessen.[96]
Testet är utformat efter metoden att originalprogrammet bryts ned i mindre delar, för att sedan ur varje del skilja ut de element som är icke- skyddade. Därefter utförs en jämförelse mellan originalprogrammet och det påstådda intrångsprogrammet.[97]

7.4.2.3.1 Abstraktionssteget.[98]

Abstraktionssteget är ingen egentlig nyhet. En motsvarande abstraktion applicerades på ett skådespel i en dom från 1930 av auktoriteten judge Learned Hands.[99] Målet rörde ett påstått intrång i en pjäs om en judisk och en irländsk familj. Efter att ha utfört abstraktion på pjäsen, där varje episod bryts ned i mindre delar, blir resultatet slutligen kärnan av pjäsen, dvs den själva oskyddade idén. Då domstolen, till skillnad från Whelan- domen, konstaterade att ett program kan bestå av ett flertal idéer, innebär det första steget att ”bryta ned” programmets beståndsdelar. Med utgångspunkt i källkoden, sker en form av teoretisk reverse engineering i många steg. Programmets helhet dissekeras till en början i sina moduler. Proceduren upprepas med varje modul, där varje del ersätts med en beskrivning av vilken funktion delen utför. Processen utförs därefter stegvis tills man slutligen har en generell beskrivning av vad programmet gör. Beroende på vilken nivå i processen man befinner sig kommer delarna att vara olika detaljerade. Ju högre i processen man befinner sig, desto mer detaljerad kommer nivån att vara. Processen går helt enkelt ut på att identifiera varje del av programmets funktion.

---

[95] Computer Associates v. Altai, s705.
[96] Termen översatt norska termen ”tretrinnsprosessen”, Se Wagle/Ödegaard s348.
[97] Computer Associates v. Altai, s706.
[98] Computer Associates v. Altai, s706f.
[99] Nichols v. Universal Pictures Co., 45 F. 2d 119, 121 (2d Cir. 1930).

7.4.2.3.2 Filtreringen.[100]
Nästa steg i processen som utförs, filtreringen, syftar till att skilja de skyddade delarna från de oskyddade elementen i programmet. På så sätt kommer själva programmets skyddsomfång att utkristalliseras. Varje abstraktionsnivå undersöks för sig själv och man skapar ett hierarkiskt system där de oskyddade elementen skiljs ut. Den högsta nivå i det hierarkiska systemet kommer bestå av en abstrakt idé men kommer att kristalliseras vidare till mer detaljrika utformningar. Beroende på i vilket steg i abstraktionsprocessen man befinner sig, kommer en specifik del av programmet vara beståndsdelen i en mer omfattande del på en högre nivå, samtidigt som den kommer att vara utgångspunkten för mer detaljerade programdelar på en lägre nivå. För varje programdel på varje abstraktionsnivå ställs därefter frågan om denna kan var utformad på ett annat sätt. Man försöker att hitta den bakomliggande idén bakom utformningen av programdelen på det konkreta sättet. Därefter sker den verkliga värderingen huruvida programdelen kunde utformas på ett annorlunda sätt, eller om utformningen styrdes av andra faktorer. De aktuella faktorerna är element dikterade av effektivitet, yttre faktorer och fria element.

a) Element dikterade av effektivitet.[101]
Domstolen konstaterade att om det i huvudsak endast finns ett sätt att uttrycka en idé, går det inte att separera utformningen från idén och skydd kan inte åtnjutas. Vid applicering av effektivitetshänsyn på själva programkoden är detta ett mindre problem. Den springande punkten i detta fall var dock i vilken utsträckning skydd kunde åtnjutas till programmets icke- litterära delar, dvs utan en direkt jämförelse mellan koderna. Liksom när undersökning om programkoden är dikterad utifrån effektivitetsaspekter, skapar datorprogrammeringens ovan nämnda inneboende prägel av effektivitet vissa problem. Även strukturen på programmet präglas av viljan att skapa ett så effektivt men framförallt minnessparande program. Ju mer effektiv strukturen är desto närmare kommer man den bakomliggande idén. Domstolen ansåg dock att motsvarande test kunde göras på programmets struktur och uttryckte att en bedömning måste göras

> ”whether the use of this particular set of modules is necessary efficiently to implement that part of the program´s process”.[102]

Om situationen är sådan att skaparen endast har ett begränsat antal möjligheter att strukturera en programdel, innebär det att denna är en direkt förlängning av den bakomliggande idén av programdelen. Här kan en

[100] Computer Associates v. Altai, s707.
[101] Computer Associates v. Altai, s707ff.
[102] Computer Associates v. Altai, s708.

jämförelse göras med ovan refererade Sybordsdomen, där domstolen kom fram till att korgens placering var dikterad av effektivitetshänsyn och därför var en direkt förlängning av idén.

b) Element dikterade av externa faktorer och fria element.[103]
Vid datorprogrammering är det näst intill omöjligt att skapa ett nytt program utan att använda sig av källkod som förvisso kan ha verkshöjd men ändå anses vara allmänt gods, kallade fria element eller public domain. Domstolen konstaterade att detta även gäller själva strukturen på programmet. Anledningen till uppkomsten av dessa fria element är beroende av ett antal skäl. Dessa kan vara att datorn som programmet skall operera på är mekaniskt beroende av vissa tekniska standarder hos programmet, annars fungerar det inte. Likaså kräver programmet en viss struktur för att vara kompatibelt med andra program. Dessutom finns det inom mjukvaruindustrin allmänt accepterade programmeringsstandarder vilka accepteras som fria på marknaden. Här finns det ett visst mått av självändamål hos programtillverkarna. Vid skapandet av ett program finns det givetvis ett egenintresse att programmet skall kunna användas på de flesta standarddatorer liksom att det är kompatibelt med andra programtillverkares program, särskilt de mest utbredda. Domstolen måste således filtrera ut de strukturer i programmet som är dikterade av dessa externa faktorer, vilka inte anses utgöra intrång i ett programs struktur.

7.4.2.3.3 Jämförelsen.[104]
Efter filtreringsprocessen är avslutad finns kärnan av den skyddsvärda strukturen kvar. De påstådda plagierade delarna av programmet jämförs med de filtrerade delarna för att undersöka huruvida intrång har skett i originalprogrammets struktur.
Efter att ha genomfört alla stegen i trestegsprocessen, kom domstolen fram till resultatet att OSCAR 3.5s struktur inte utgjorde någon kränkning av ADAPTERs SSO.[105]

7.4.3 Åtnjuter SSO även skydd i Europa?

Den mest intressanta aspekten med Altai- fallet är i vilken utsträckning de icke- litterära momenten i ett datorprogram åtnjuter skydd. I Whelan- fallet konstaterade domstolen att den övergripande bilden av ett datorprogram, kodtexten, strukturen av programmet men även de genererade ut- effekterna, skulle ligga till grund för den direkta jämförelsen mellan programmen. I Altai- fallet uttrycks skyddet för de icke- litterära elementen mycket vagt. Domstolen konstaterade att ”To be frank, the exact contours of copyright protection for non- literal program structure are not completely clear”.[106] Domstolen överlämnade istället klargörandet av gränserna för skyddet till framtida avgöranden. Viss kritik riktades även mot domstolen att något

[103] Computer Associates v. Altai, s709f.
[104] Computer Associates v. Altai, s710ff.
[105] Computer Associates v. Altai, s721.
[106] Computer Associates v. Altai, s712.

egentligt klargörande av rättsläget inte hade skett i och med domen, utan att den tvärt om istället skapade alldeles för många nya frågetecken.[107] Med trestegsprocessen samt med hänvisning till att datorprogram är litterära verk konstaterades dock att de icke- litterära elementen i ett program kan åtnjuta skydd.

Frågan som väcks med bakgrund av Altai- fallet är om skydd för de icke-litterära elementen i ett program även kan åtnjutas i Sverige och Europa. EG- direktivet tar inte ställning till det och det nämns heller inget om ett sådant skydd i de svenska förarbetena, utan spekulationer får ske med bakgrund av direktivtexten. Art 1.2 uttrycker att datorprogram åtnjuter skydd ”in *any* form” [min kursivering], samtidigt som hänvisning sker till definitionen av datorprogram som stadgas i Bernkonventionen. Att flödesscheman numera även skyddas som datorprogram, tyder på att det traditionella synsättet att datorprogram är relaterade till programkoden är övergiven. Begreppet har därför vidgats betydligt och verkar ge öppningar för att skydd kan åtnjutas för fler uttrycksformer. [108] Frågan grundar sig helt enkelt i vad som kan läggas i begreppet datorprogram.

För att skydd för programmets form även skall kunna åtnjutas måste till att börja med en viss psykologisk spärr övervinnas; man måste se bortom de litterära aspekterna i programmet. Förarbetena definierar just datorprogram som käll- och objektkod, dvs de konkreta litterära elementen i ett program. Programmens funktionsbaserade karaktär innebär dock att strukturen i ett program i ytterst stor utsträckning svarar mot hur den avsedda funktionen skall uppnås. Effektiviteten i ett program baserar sig dessutom i allra högst grad på hur programmet är strukturerat. Det skyddsvärda i ett program sträcker sig därför i stor utsträckning bortom de litterära elementen i ett program. Strukturen i ett program hör även samman med kvaliteten på programmet. Ovan nämnda resonemang om verkshöjdsbedömningen blir återigen aktuellt. Är kvalitetsfaktorn i ett program en aspekt som påverkar skyddsomfånget resulterar det i att det är att lättare acceptera de icke-litterära elementen i ett program som skyddsvärda. Kvaliteten i ett program visar sig ofta i hur programmet är strukturerat och organiserat och därför en anledning att anse de icke- litterära elementen som en förlängning av skaparens uttryck och därför göra de skyddsvärda.
Det finns även samhällsekonomiska aspekter på synsättet. Datorprogrammen utgör en verkskategori som årligen omsätter ofattbara summor. Att även definiera de icke- litterära aspekterna i ett program som skyddsvärda, stärker ytterligare incitamenten hos tillverkarna att investera medel i industrin.

[107] Branley Thomas H., Is copyright protection for ”structure, sequence and organization” dead? http://www.lgu.com/cr43.htm 990217.
[108] Plogell Michael, s37f.

7.4.4 Skyddsomfånget.

Att skydd även skall åtnjutas för strukturen i ett program, skapade efter Altai- domen viss oro, både hos programleverantörer och även inom doktrinen.[109] Vad som åsyftades var att risk för monopolställning för strukturerna i ett program föreligger. Det skulle kunna hindra oberoende programtillverkare att kunna skapa kompatibilitet både med marknadsledande program och med hårdvaran. Det är dock viktigt att skilja på ett konstaterat skydd och motsvarande skyddsomfång för skyddsobjektet. Ett konstaterat skydd behöver nödvändigtvis inte innebära monopolställning, så länge skyddsomfånget är snävt. Dekompilering är tillåtet i UrL, under ovan nämnda omständigheter, för att skapa samverkan mellan program. I det avseendet spelar skyddsomfånget för strukturen ingen roll, programinnehavaren är fri att kopiera originalprogrammets struktur. Frågan är om dekompileringsbestämmelsen även omfattar att skapa kompatibilitet mellan hårdvara och mjukvara. Ett skydd för strukturen i ett program tillsammans med ett förbud att använda den dekompilerade informationen för att skapa samverkan med en icke- kompatibel hårdvara, skulle riskera monopolställning hos hårdvarutillverkare. Är skyddsomfånget dessutom brett beroende på att kvaliteten i strukturen är originell, är problemet än mer påtagligt. Det råder delade meningar om lydelsen i 26h§ UrL omfattar både hård- och mjukvara.[110] En viss självreglering finns dock i branschen. En hårdvarutillverkare avser att så många program som möjligt skall kunna användas på de tillverkade maskinerna.
I fall som inte hänförs till avseenden att skapa kompatibilitet måste skyddsomfånget bedömas i varje fall. Ovanstående resonemang rörande verkshöjden blir därför aktuellt. *Om* kvalitetsaspekten omfattas av momenten som skall bedöma skyddsomfånget, åtnjuter programstrukturer av hög kvalitet större skydd.

7.4.5 Är trestegsprocessen tillämpbar även i Sverige?

Metoden har fått genomslagskraft i USA och följts av ett antal senare fall. Domen har dock blivit utsatt för en hel del kritik, dels med avseende på tillämpningen av modellen, dels med avseende på att några tydliga riktlinjer för skyddet för de icke- litterära aspekterna av ett program inte gavs.[111] Vad gäller tillämpningen av trestegsprocessen på datorprogram framstår den, med bakgrund av Altai- fallet, som en bra modell att tillämpa på datorprogrammens icke- litterära element. Att tillämpa den direkt på programkoden är inte nödvändigt, då en direkt jämförelse mellan programmens koder är betydligt enklare att genomföra. Direkta jämförelser i metoden kan man göra både med Sybordsdomen och hovrättsfallet, refererade ovan. De domarna konstaterade just att element som är dikterade

[109] Liew Hui- Ming, Sequerah Andre Sean, Programmers and The Law- Final Report, http://www-dse.doc.ic.ac.uk/~nd/surprise_95/journal/vol4/hml/report.html 990217.
[110] Se Plogell Michael s56 jämfört med Rosén Jan s38.
[111] Branley Thomas H., Is copyright protection for ”structure, sequence and organization” dead? http://www.lgu.com/cr43.htm 990217.

av effektivitet och yttre faktorer inte kan åtnjuta skydd. Problemet är att dock, som ovan konstaterats i kapitlet om kravet på verkshöjd, att effektivitet och kvalitet till stor del präglar programmeringen. Själva filtreringen är förvisso genomförbar men skyddsobjektet måste först vara definierat i bemärkelsen att effektivitetsaspekten i ett program inte konsekvent drabbas i processen. En direkt jämförelse med Sybordsdomen, kan enligt mig därför inte generellt göras.[112] Om kvaliteten på ett program spelar in i skyddsbarheten, måste alltså först konstateras genom EG- rättslig praxis. Sybordsdomen konstaterande att de element som är en direkt förlängning av verkets idé inte åtnjuter skydd, går även igen i Altai- domen. På samma sätt är modellen applicerbar för att skilja de fria elementen i ett programs struktur. Med avseende på det genomslag modellen fått i USA med anledning av att det idag inte finns någon bättre metod att jämföra de icke- litterära momenten, anser jag att metoden även skulle kunna tillämpas i Sverige idag, med ovan nämnda hänsynstaganden.

7.4.5.1 Praktiska problem.

En generell modell som är tillämpbar på dagens programmeringstekniker kan, med avseende på den snabba tekniska utvecklingen inom området, vara svårare att tillämpa på kommande tekniker.[113] Det är därför fel att påstå att trestegsprocessen är den slutliga lösningen på jämförelsen mellan datorprogram skrivna i olika programspråk, där intrång påstås föreligga i programmets struktur och organisation. Trestegsprocessen kommer därför säkert att genomgå förändringar, eller rent av att överges, i takt med utvecklingens gång. Judge Boudin, nedan i Lotus- fallet, uttryckte följande i samband med datorteknikens intåg på den upphovsrättsliga arenan att ”Applying copyright law to computer programs is like assembling a jigsaw puzzle whose pieces do not quite fit.”.[114]

7.4.6 Användargränssnittet

7.4.6.1 Inledning

Det finns ytterligare en aspekt av de icke- litterära komponenter som ur en konkurrensmässig synvinkel har fått en större betydelse ju mer sofistikerade datorprogrammen blir - användargränssnittet.
Användargränssnittet måste först och främst skiljas från det tekniska gränssnittet. Det senare medför kompatibilitet mellan dels datorns hårdvara, dels med andra program. Det tidigare möjliggör en ömsesidig kommunikation mellan programmet och användaren. Samtidigt består användargränssnittet av två komponenter. Den första är själva grafiken som återspeglas på bildskärmen och den andra är de operationer som kan utföras på skärmen, ofta med tangentbordet som hjälp. Ett exempel är F1- tangenten på tangentbordet som i de flesta populära program är kortkommandot för hjälpfunktionen. Användargränssnittet underlättar kommunikationen mellan programmet och användaren, samtidigt som det ger möjlighet för densamme

[112] För avvikande uppfattning se Wagle/Ödegaard s349.
[113] Plogell Michael, s28.
[114] Lotus v. Borland s820.

att styra programmets funktioner. Trots att den bakomliggande koden till användargränssnittet utgör en mindre del av programmets totala kod, utgör det en mycket viktig aspekt att lära användaren att hantera programmets funktioner. Vid framtida programinköp är det rimligt att anta att, för att underlätta bruket av programmet, ett användargränssnitt som kunden är van vid, är en väsentlig faktor vid valet av program. Kunden slipper således besväret med att på nytt lära känna miljön med dess funktioner och kan istället snabbt börja använda programmet.

Precis som liknande programfunktioner kan genereras med olika källkodstruktur eller programspråk, kan även totalt olika programspråk generera identiska skärmbilder och operationellt betingade gränssnitt. Uppkomsten av konflikter på området är därför uppenbar och har lett fram till ett antal rättsfall i USA och genererat en omfattande diskussion i doktrinen. Jag hade tänkt att nedan redogöra för det viktigaste rättsfallet i USA, ett fall som prövades i landets högsta domstol. Därefter redogör jag för de argument som ligger bakom utgången i målet och spekulerar i om utgången i målet är relevant för svenska och europeiska förhållanden.

7.4.6.2 Lotus- fallet

Ett mycket uppmärksammat fall avgjordes efter många års prövning i amerikansk domstol av landets högsta domstol, Lotus v. Borland.[115] Det måste nämnas att det nedan är refererat med utgångspunkt i apellationsdomstolens dom. I högsta domstolen förklarade en av de sju domarna sig inhabil, medan de andra sex var oeniga. Apellationsdomstolens dom stadfästes utan att de kvarvarande sex domarna offentliggjorde sitt beslut.

7.4.6.2.1 Bakgrund.[116]

Lotus konstruerade i mitten av 1980- talet ett kalkylprogram som fick stort genomslag på mjukvarumarknaden, Lotus 1-2-3. En stor del av programmets popularitet berodde på programmets logiska användargränssnitt, som gjorde det möjligt för användaren att med hjälp av kortkommandon eller enkla pilförflyttningar styra programmets funktioner. Gränssnittet var uppbyggt likt ett hierarkiskt träd i 50 menyer och submenyer. Borland, ett konkurrerande företag, lanserade 1987 ett kalkylprogram, Quattro, vars menyuppbyggnad framstod som en klon av Lotus 1-2-3. Viktigt att ha i åtanke är att den bakomliggande koden som genererade de båda användargränssnitten var helt olika. Borland hade inte ens haft tillgång till Lotus källkod, utan hade kopierat gränssnittet utifrån bildskärmen och funktionerna på Lotus 1-2-3. Borland medgav att man faktiskt hade kopierat Lotus hierarkiska gränssnitt men menade att Lotus inte kunde åtnjuta skydd för det samma. Det domstolen hade att ta ställning till var i viken utsträckning skydd kunde åtnjutas för användargränssnittet.

---

[115] Lotus Development Corp. v. Borland International, Inc. 49 F 3d 807 (1 st Cir. 1995).
[116] Lotus v. Borland, s809ff.

Med denna utgångspunkt är situationen annorlunda än i det ovan refererade Altai- fallet. Den domen handlade ju om i vilken omfattning de icke-litterära delarna av ett program kunde åtnjuta skydd, något som domstolen prövade med trestegsprocessen. Även om domstolen i det här fallet ansåg att denna metod var utmärkt för att pröva skyddet i det avseendet, ansåg den att metoden inte var applicerbar i detta fall.[117] Istället hade domstolen att ta ställning till huruvida det hierarkiska användargränssnittet överhuvudtaget kunde skyddas.

7.4.6.2.2 ”Method of operation”
Enligt Copyright Act §102 (b), ovan citerad, skyddas inte de metoder som ligger bakom uttryckandet av ett verk. Domstolen menade att det hierarkiska systemet med menyer, vilka var ett medel att styra programmets funktioner, just var sk ”methods of operations”.[118] En användare behövde inte ta del av det sätt på vilket menystrukturen var uttryckt för att kunna operera programmet och därmed var även menystrukturens uttryck en del av själva metoden. Domstolen jämförde den funktionsbaserade menystrukturen med en videobandspelare.[119] På samma vis som en videospelares ”play-funktion” aktiveras med att trycka på en knapp, är kortkommandot ”P” för ”Print” inget annat en metod för att kunna använda spelaren på avsett sätt. Lotus 1-2-3´s typsnitt kunde ha utformats annorlunda, dvs gränssnittets grafik kunde vara mer originellt men även om detta hade skett skulle själva kortkommandona vara en metod för att kunna använda programmet. Därför behövde domstolen inte vidare undersöka om metoden kunde ha varit designad på ett annat sätt. Sålunda kom domstolen fram till ett annat resultat än vad distriktsdomstolen gjorde och talan ogillades.

7.4.6.3 Det visuella gränssnittet
I Lotus- fallet var användargränssnittet inte speciellt visuellt uttrycksfullt. Klart är dock att även användarmiljön i form av det skärmbilden genererar är en stark faktor i att knyta till sig köpare, framför allt för kommande inköp av uppgraderade versioner av program. Med tanke på att även den visuella miljön kan genereras med olika programspråk och programinstruktioner, uppkommer även frågan i vilken utsträckning de icke- litterära aspekterna i form av bildskärmen som koden genererar skyddas. Vid denna frågeställning måste man ha i åtanke att datorprogrammet ses som ett verk, dvs käll- och objektkod men även i alla de former det framträder. Således måste en originalitetsbedömning ske utifrån dessa aspekter.[120] Detta är även uttryckt i direktivet för datorprogram Art 1.2.. Kränkning kan därför föreligga om skärmens utseende är originellt, även om den bakomliggande koden är annorlunda.
Detta är dock ett generellt påstående som grundar sig på den grundläggande definitionen om vad som omfattas av begreppet datorprogram, dvs instruktioner till datorn för att uppnå ett specifikt resultat, en funktion eller

[117] Lotus v. Borland, s814f.
[118] Lotus v. Borland, s815ff
[119] Lotus v. Borland, s817.
[120] Bender Hanne, NIR 1997 s75f.

uppgift. Bilder, musik etc, kan dock även vara lagrade i datorn i form av filformat. Bilder kan scannas in för att till exempel visas som en pausfunktion. Bilder kan även skapas med designprogram såsom CAD-program och även de lagras i datorn. Dessa typer av verk faller per definition inte in under begreppet datorprogram, då de inte utgör instruktioner till datorn för att uppnå ett specifikt resultat och skyddas därför under respektive verkskategori. Ett aktuellt exempel är de i dag mycket utbredda musikfilerna lagrade i filformatet MP3. Musiken i filerna åtnjuter skydd under kategorin musikaliskt verk i UrL 1§ 1p medan programmet som återspelar filerna i datorn, exempelvis Winamp, härleds till kategorin datorprogram.

Bildskärmens aktualitet uppkom i USA på 1970- talet, då videospelens expansion påbörjades. Många av fallen rörde det då mycket populära spelet ”Pac- Man”, som blev utsatt för otaliga kloningar.[121] Bildskärmen kan dock registreras som ett audiovisuellt verk i USA, då själva skärmutseendet åtnjuter skydd men inte den bakomliggande koden.[122]
En originalitetsbedömning måste ske utifrån skärmbilden. När det gäller bruksprogrammen resulterar det i, med Lotus- fallet i åtanke, att skyddet är relativt snävt. För dessa program spelar funktionaliteten en stor roll och den svåra gränsdragningen av vad som är funktionellt betingat med skärmdialogen och vad som är utformning gör att det funktionella aspekterna nog i många fall tar överhanden. Ur denna aspekt kritiserades Lotus- fallet av många i USA.[123] Författarna till artikeln menar att då domstolen ansåg att det funktionspräglade användargränssnittet var oskyddat, uttrycktes detta alldeles för generellt. Följden skulle bli att alla former av användargränssnitt, *oavsett hur det är uttryckt*, inte kan åtnjuta skydd p.g.a. sin funktionella karaktär. Samtidigt menar de, att med utgångspunkt i domstolens resonemang, innebär det att den underliggande koden till gränssnittet är oskyddad, då den genererar det oskyddade gränssnittet. Jag anser att författarna har fel i sin kritik. Kritiken grundar sig istället på den osäkerhet som har skapats i USA under åren med otydliga avgöranden i vad som är ett exemplar av datorprogrammet. Har man istället klart för sig att datorprogrammet skall anses som ett verk, som det gör i Sverige och skyddet för datorprogrammet även omfattar det som genereras, är kritiken inte nödvändig. Kritiken kan ur en del avseenden vara berättigad. Utvecklingen har gått snabbt i USA beträffande användargränssnittet, från att ha ett relativt brett skydd till ett väsentligt snävare skyddsomfång. Ur ett svenskt perspektiv skall man i detta avseende inte stirra sig allt för blind på hur situationen i USA har utvecklat sig. Det tidigare breda skyddet i USA berodde bl a på att landet, till skillnad från Sverige, saknar en federal marknadsföringslag. Det fanns därför större incitament att skydda användargränssnittet i en vidare bemärkelse. Jag kommer att demonstrera det med en norsk dom fån 1989, som bekräftar min slutsats.

---

[121] Se ex. Midway Manufacturing Co v. Strohon 564 F. Supp 741 (E.D. Ill. 1983).
[122] Schmidt Per Håkon, Teknologi og immaterialret, s185f.
[123] Ses bl a Goldberg David Morton mfl, Berkeley Technology Law Journal, http://www.law.berkeley.edu/journals/btlj/lvb/dec-amic.html 990217.

7.4.6.3.1 Aladdin- domen

Den sk Aladdin- domen rörde påstått intrång i ett konverteringsprogram.[124] De båda programmen var konkurrerande och käranden påstod att det visuella uttrycket i svarandes program var så likt kärandes att det stred mot god marknadsföringssed, enl norska MFL 1§. Fallet avgjordes inte efter upphovsrättslig lagstiftning, vilket heller aldrig var på tal. I stället konstaterade domstolen att för de visuella element på skärmbilden som *inte* var av funktionell karaktär, finns det en plikt att undgå förväxling. Likaså konstaterade rätten att eftergörning av de branschspecifika koncept som inte ges uttryck i designen, inte är olovliga att nyttja.

Domen är intressant ur två aspekter. För det första bekräftar domen att utformning som är en följd av dess funktionella karaktär inte kan åtnjuta skydd, dvs motsvarande resonemang som fördes i Lotus- fallet. Det innebär att rullgardinsmenyer, ikoner etc ej kan åtnjuta skydd. Detta är dock inget generellt stadgande. Är användargränssnittet originellt utformat kan givetvis skydd åtnjutas. Windows fönstersystem kan ses som pionjär inom menystrukturen för operativsystem. Det kan därför tänkas att det åtnjuter upphovsrättsligt skydd.[125] För det andra visar domen att slavisk kopiering kan strida mot marknadsföringslagens regler även om utformningen av användargränssnittet inte har sådan originell karaktär att det skyddas upphovsrättsligt, beroende på otillräcklig verkshöjd eller att gränssnittet har genererats av olika programspråk. Problemet är att det i Norge ges ett starkare skydd mot otillbörlig efterbildning än i Sverige. I 8§ MFL finns dock ett visst skydd mot vilseledande efterbildningar vid marknadsföring. Sverige har i viss mån i större utsträckning förlitat sig det immaterialrättsliga skyddet vid plagiat men den norska domen känns ändå aktuell för svenska förhållanden.[126]

[124] Strömmens herredsrett 28/6 1989, NIR 1992 s133.
[125] Plogell Michael, s51.
[126] Koktvedgaard Mogens, Levin Marianne, s74.

8. Utvecklingen i framtiden

Som nämndes i inledningen har UrL gjorts teknikoberoende, detta för att vara applicerbar på nya tänkbara framtida verkskategorier. Utvecklingen inom mjukvaruindustrin fortgår med en rasande takt och de program som idag kategoriseras som pionjärprogram, är nästa dag förlegade. Under den relativt korta period som datorprogram har varit upphov till tvister i USA, har begrepp och metoder avlöst varandra i takt med att nya programmeringstekniker har utvecklats. Idag skrivs de flesta program i fjärde generationens programmeringsspråk men program som Visual Basic ger exempel på att den kommande generationens programspråk inom kort kommer att vara standard. Femte generationsspråkens inträde kommer att innebära att fler instruktioner ersätts med färre programmeringskommandon. Samtidigt kommer nya strukturer inom programmeringen utvecklas. Idag är t ex Java- tekniken väldigt utbredd inom IT- området. Java är ett språk som kraftigt skiljer sig mot de traditionella programspråken, då det är ett sk plattformsbaserat språk. Strukturen är därmed vitt skilt från exempelvis Pascal som tillämpar procedurformer. Den ovan redogjorda trestegsprocessen blev som nämnt utsatt för en del kritik för att i alldeles för stor utsträckning endast vara tillämpbar på det avsedda programmet och inte vara någon generell teknik. Kritiken är kanske till viss del berättigad men å andra sidan speglar den rådande verkligheten på området. Det upphovsrättsliga skyddet för datorprogram kommer därför även i framtiden att sättas på prov.

9. Sammanfattande slutsatser

Datorprogrammens komplexa natur och dess inneboende skillnader mot traditionella litterära verk har satt upphovsrätten på prov. Det har visat sig att det har varit svårt att applicera grundläggande upphovsrättsliga begrepp på datorprogrammen. Den obefintliga EG- rättsliga praxisen på området samt förarbeten med oklara riktlinjer innebär att några klara direktiv inte finns att tillgå för skyddet för datorprogram. Jag har försökt att redogöra för den praxis som utvecklats främst i USA för att undersöka om paralleller kan göras med svenska och europeiska förhållanden.
I de avseenden där direkta jämförelser kan göras mellan påstådda kränkta program och kopior är ofta element som hjälptexter, variabelnamn och infogade delar till stor hjälp för att påvisa om intrång har skett. I de fall då programspråken inte är närbesläktade blir situationen mer svårbedömd. Översättningssituationen i UrL 4§ 2 st. blir inte alltid tillämplig utan överväganden får göras i varje konkret fall. I samband med plagiatvärderingen har jag även undersökt vilken betydelse de genererade effekterna har för bedömningen. Bakom snarlika användargränssnitt kan dölja sig direkta skillnader i programkoden. Med infallsvinkeln att datorprogram skyddas som ett verk blir situationen mer lättöverskådlig. Alla dess framträdelseformer är därmed skyddade oberoende av den bakomliggande koden om de traditionella upphovsrättsliga kriterierna är uppfyllda.
I samband med de amerikanska rättsfall jag har redogjort för, har även frågan uppkommit huruvida programmets struktur och organisation, dvs de icke- litterära elementen i ett program kan åtnjuta skydd. Trestegsprocessen har utvecklats för att identifiera eventuella skyddsvärda strukturer. Med bakgrund av det vaga EG- direktivet och de svenska förarbetena som fokuserar sig på programkoden, måste slutsatser dras utifrån det egentliga skyddsvärda i programmet. Med tanke på att programmeringen i allt större utsträckning blir mer objektorienterad tycker mig jag kunna dra slutsatsen att även strukturer i datorprogram kan åtnjuta skydd.
Framtidens metoder för att identifiera eventuella plagiat måste förfinas takt med programmeringens utveckling. Det viktiga i sammanhanget är dock att skyddsobjekten är identifierade.

Litteraturförteckning

- Andersen Mads Bryde, Laerebog i EDB- ret, Jurist- og ökonomförbundets Förlag, Köpenhamn 1991.
- Koktvedgaard Mogens, Levin Marianne, Lärobok i Immaterialrätt, 5 uppl., Norstedts Juridik, Sthlm 1997.
- Levin Marianne, Vennebog til Mogens Koktvedgaard, Nerenius & Santérus förlag, Sthlm 1993.
- Rosén Jan, Swedish Software Law as related primarily to the EC Directives, Juristförlaget Stockholm 1995.
- Schmidt Per Håkon, Teknologi og immaterialret, GAD, Köpenhamn 1989.
- Stensaasen Tarjei, Rettslig vern av EDB- programmer og databaser, TANO Oslo 1987.
- Plogell Michael, Immaterialrättsliga aspekter på datorprogram, Juristförlaget, Sthlm 1996.
- Wagle Mediaas Anders, Ödegaard jr Magnus, Opphavsrett i en digital verden, Cappelen Oslo 1997.

Artiklar

- Bender Hanne, Ophavsret til brugergraenseflader ”Look and Feel”, NIR 1997 s69.
- Bing Jon, Opphavsret og ny informasjonsteknologi: Noen spredte notater, NIR 1995 s595.
- Coleman Susan E, The State of Software Copyright Protection in the United States, NIR 1988 s 111.
- Karnell Gunnar, Upphovsrätt till datorprogram, NIR 1984 s183.
- Nordell Per Jonas, Om skyddsomfång vid upphovsrättsligt skapande, särskilt med hänsyn till skapande med hjälp av datorteknik, NIR 1991 s369.
- Nordell Per Jonas, Dubbelskapande i teori och praktik, NIR 1995 s630.
- Seipel Peter, Datorprogrammens rättsskydd NIR 1973 s143.
- Ulmer E, Kolle G, Copyright Protection of Computer Programs, IIC No 2. 1983 s159.

Förarbeten

- SOU 1956:65.
- SOU 1985:51.
- Prop 1988/89:85.
- Prop 1992/93:48.

Rättsfallsförteckning

Amerikanska rättsfall
- Baker v. Selden, 101 U.S. 99 (1879)
- 201 U.S. (1908), White- Smith Music Pub. Co v. Apollo Co.
- Nichols v. Universal Pictures Co., 45 F. 2d 119, 121 (2d Cir. 1930).
- Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 219 U.S.P.Q.113 (3d Cir. 1983).
- Q- Co. Industries, Inc. v. Hoffman, 625 F.Supp. 608 (S. D. N. Y. 1985).
- Whelan Associates v. Jaslow Dental Laboratory Inc., 797 F. 2d, 1222 (3d Cir. 1986).
- Feist Publications, Inc. v. Rural Telephone Service Company, Inc. Supreme Court of the United States, No. 89-1909, 27. March 1991.
- Computer Associates International, Inc. v. Altai Inc., 982 F. 2d, 693 (2d Cir. 1992).
- Lotus Development Corp. v. Borland International, Inc. 49 F 3d 807 (1 st Cir. 1995).

Fall från högsta domstolen
- NJA 1938 s497
- NJA 1990 s168
- NJA 1990 s499
- NJA 1994 s74
- NJA 1996 s79

Hovrättsavgöranden
- HovR f. Västra Sverige 19/11 1987.

Australiska rättsfall
- Apple Computer Inc. and Apple Computer Australia Pty Ltd v. Computer Edge Ltd and Michael Suss. Fed Ct. of Australia, NSW Distr. Reg. Gen. Div. 7/12 1983.

Franska rättsfall
- Apple II v. Golem, Trib. Gr. Inst. Paris 14/6 1983.
- Soc. Babolat Mailott Witt c. P., Gaz Pal. 3/3 1983 s117.

Danska rättsfall
- UfR 1993 s17

Norska rättsfall
- Rt. 1962 s964
- Oslo byrett 18/3 1986

Tyska rättsfall
- Inkasso- domen, Bundesgerichthof I ZR 52/83 850509.

URL- adresser

- Byer, David J, Gall, Nicholas L., Developments in software law, Part II: Copyright protection of non- literal aspects. http://www.tht.com/204875.htm 990217.
- Goldberg David Morton mfl, Berkeley Technology Law Journal, http://www.law.berkeley.edu/journals/btlj/lvb/dec-amic.html 990217.
- Liew Hui- Ming, Sequerah Andre Sean, Programmers and The Law- Final Report. http://www-dse.doc.ic.ac.uk/~nd/surprise_95/journal/vol4/hml/report.html 990217.
- Russo Jack, Nafsiger, Jamie, Software ”Look and feel” protection in the 1990´s. http://www.computerlaw.com/lookfeel.html 990217.
- Branley Thomas H., Is copyright protection for ”structure, sequence and organization” dead? http://www.lgu.com/cr43.htm 990217.

Referenslitteratur

- Andersen Mads Bryde, EDB og Ansvar- Studier i edb- erstatningsrettens beskrivelseproblematik, Jurist og Ökonomforbundets forlag, Köpenhamn 1989.
- Andersen Mads Bryde, Ophavsretten og den nye teknologi, Information og kommunikation- digitalisering og superhighways, NIR 1995 s616-
- Bing Jon, Opphavsrett og EDB, Complex 2/85, Universitesforlaget, Olso 1985.
- Butler John H., Pragmatism in Software Copyright: Computer Associates v. Altai. http://jolt.law.harvard.edu/low/abstracts/6hjolt183.html
- Lindberg Agne, Westman Daniel, Praktisk IT- rätt, Norstedts Juridik, Sthlm 1997
- Schricker Gerhard, Farewell to the ”Level of Creativity (Schöpfungshöhe) in German Copyright Law? IIC 1995 s41.
- Maiorana David M., Priviliged Use: Has Judge Boudin suggested a viable means of copyright protection for the non- literal aspects of computer software in Lotus Development Corp. v. Borland International? http://www.wcl.american.edu/pub/journals/lawrev/MAIORTXT.htm 990217
- Soma John T., A comparison of German and U. S. Experiences in Software Copyrights, IIC No 6 1987 s751.
- Tysver Daniel A., Bit Law Source Cases: Feist, http://www.bitlaw.com/source/cases/copyright/feist.html 990217.
- Gelman Steven J., Keeping Computer Associates v. Altai Current with Changes in Computer Program Development, http://ourworld.compuserve.com/homepages/sjgelman/altaitext.html 990217.