Softwarepatenter i USA

David Alexander Bjerrum Petersen
Studienummer: 20114600
Efterår 13/12-2013
Tegn: 34379.
Bacheloropgave.
dabp1@hotmail.com
Studieordning: BA 2010.
Institut for kultur og samfund - Idéhistorie

Indholdsfortegnelse

Softwarepatenter i USA – Bacheloropgave
David Alexander Bjerrum Petersen, 20114600

Abstract

Even though software isn't mentioned in the patent law of the U.S, the first software patent got litigated in 1981, in the case of Diamond v. Diehr. Since then, the patent application's and litigation's have increased every year, and simultaneously the uncertainty of software patent's validity has increased.

This paper is a review of how software patent's where litigated through the three supreme court cases of Gottschalk v. Benson, Parker v. Flook and Diamond v. Diehr, and why it is a problem that software becomes patentable through the court and not through the Congress. The case of Apple v. Samsung points out that the patent system of U.S no longer is sufficient to the needs of the new industries like software, because software industries do not work as, for instance, the car industry, and therefore the principle for innovation is different.

This paper concludes that the software patent's are causing problems, because there wasn't any purpose or idea with making software patentable in the first place, and the way to fix it, is by trying to understand how software innovation works and then amend these ideas to the patent system.

Indledning

Da elektronikgiganten Apple i 2011 sagsøgte Samsung, for at bryde nogle af patenterne til deres iPhone, var det et tegn på en ny og mere aggressiv måde at bruge sine patenter på.

En tendens hvor der i de seneste ti år, er kommet flere og flere retssager om brud på patenter i software og elektronik industrien, selvom det ellers er normalt for elektronikfirmaer at bruge andres patenter, mod en betaling i royalties, da produkter som smartphones er så komplekse at det ikke kan lade sig gøre at lave dem uden at bryde andres patenter (Pwc 2012: p. 14).

Apple Inc v. Samsung Electronics Co. Ltd. skiller sig derfor ud, da målet med retssagen ikke var at indgå et forlig hvor der skulle betales royalties for brugen af patenter, men helt at forbyde alle de produkter fra Samsung, der bryder Apples patenter (Apple Inc v. Samsung Electronics Co., LTD 2013: pp. 3-4).

Men hvordan kan det være, at et multinationalt selskab som Samsung bruger milliarder på at udvikle og sælge produkter, uden at sikre sig, at det ikke krænker et andet firmas patenter? Burde patenter ikke skabe så stor sikkerhed om hvem der ejer retten til en funktion i et produkt, at det ikke ville kunne betale sig at føre en retssag om det, da resultatet ville være givent?

Hvis man kigger nærmere på hvilke patenter Apple har sagsøgt Samsung for at bryde, består det bl.a af softwarepatenter for softwarefunktioner som: "Portable electronic device, method, and graphical user interface for displaying structured electronic documents" (U.S Patent 7,844,915 2011; Apple Inc v. Samsung Electronics Co., LTD 2013: pp. 7-8).

Patenter jeg undrer mig over er gyldige, da de umiddelbart virker vage og upræcise, fordi de ikke indeholder detaljerede beskrivelser af, hvordan og om den patenterede opfindelse kan fungere i praksis (Ibid).

Selvom software ikke er nævnt i den amerikanske patentlovgivning, har det siden retssagen mellem Diamond v. Diehr i 1981, været mulig at patentere software under de universelle regler for patenter (Cohen & Lemley 2001: p. 9).

Jeg vil undersøge hvordan de universelle grænser for hvad der kan patenteres og hvilken rækkevide det har, har ændret sig til også at favne softwarepatenter - samt hvilken betydning denne ændring har for softwarebranchen og patentsystemet.

Mit fokus vil være på softwarepatenter, da softwarepatenter er en del af patent-krigen mellem Apple og Samsung, samtidig med, at det er et af de nyeste felter der er blevet gjort patenterbare.

Jeg har valgt at begrænse mig til USA, da de største softwarevirksomheder i verdenen har

hovedsæde i USA, samtidig med at det er et at de lande med færrest begrænsninger på software patenter - hvor man f. eks. i Europa ikke har mulighed for at patentere rent software (Forbes 2013).

Problemformulering

"Jeg vil undersøge hvordan amerikanske patenter indenfor software-industrien, bliver forsvaret som værende intellektuel ejendom og derved patenterbart, og om softwarepatenter skaber et problem for det nuværende patentsystems berettigelse gennem retssagerne mellem Apple og Samsung."

Jeg vil starte med at undersøge formålet med det amerikanske patentsystem, da det vil give en forståelse for formålet med patenter, samt hvilken patentlovgivning software skal passe ind i.
Derefter vil jeg gennemgå de tre retssager, der hver har været med til at fjerne de forhindringer i patentlovgivningen, som betød at software ikke var patenterbart.
Til sidst vil jeg gennem retssagerne mellem Apple og Samsung komme ind på de problemer der opstår med softwarepatenter i patentsystemets nuværende form, samt diskutere om der overhovedet er behov for et software patent system.

Grundlaget for USA's patentsystem

Det at kunne patentere en opfindelse fordi man var den første til at opfinde den, er ikke en konstitutionel ret ligesom ejendomsretten. Patenter er i stedet et middel som staten kan bruge til at give opfindere et incitament for at innovere og kommercialisere deres opfindelser, for at øge den sociale velfærd: ”The Congress shall have power...To promote the progress of science and useful arts, by securing for limited times to authors and inventors the exclusive right to their respective writings and discoveries” (Constitution of the United States 1789: Article One, Section 8,8).
Legitimeringen af patentlovgivningen i konstitutionen gør det muligt for kongressen at tilpasse genstandsfeltet[1] for patentlovgivningen, til at indeholde de felter som kongressen finder nødvendigt for at fremme videnskaben og innovationen – hvilket betyder at kongressen vil have hjemmel for f. eks. at indføre softwarepatenter i patentlovgivningen.
Det at patentlovgivningen ikke er låst fast af konstitutionen har samtidig betydet, at patentlovgivning har haft en del ændringer siden den første patentlovgivning i 1790, hvoraf jeg her vil komme ind på to vigtige ændringer, der har haft en betydning for den senere indførelse af softwarepatenter.

1 Frit oversat fra de amerikanske "subject of matter".

I 1793 blev den første patentlovgivning erstattet med en ny, hvor en af de vigtigste tilføjelser er definitionen af genstandsfeltet for patentlovgivningen: "... any new and useful art, machine, manufacture or composition of matter, or any new and useful improvement on any art, machine, manufacture or composition of matter," (Patent Act of 1793: Section 1).

En definition der, med undtagelsen af 'useful art' som i 1953 blev ændret til 'useful proces', stadig bliver brugt i § 101, til at definere genstandsfeltet for patenter i dag (Title 35 of the United States Code 1952: § 101).

Denne definition betød bl.a. at abstrakte idéer og derunder matematiske algoritmer, ikke kunne patenteres, da disse ikke blev tolket som opfindelser lavet af mennesker, men som opdagelser af naturgivne regler (Thomas 2008: p. 194). Da software er lavet af matematiske algoritmer betød det at software blev afvist som patenterbar genstand, da det blev kategoriseret som en abstrakt idé (Samuelson 1990: p. 1057).

Den anden vigtige ændring skete i 1836, hvor fundamentet for det nuværende patentsystem blev lavet. Ændringen var karakteriseret ved en nemmere ansøgningsproces, hvor der var tekniske eksperter til at vurdere ansøgningerne, et lavere gebyr for at sende en ansøgning, en frit tilgængelig database over patenter og en strømlining af kvalifikationskravene for godkendelser af patenter – det skulle være nemt og hurtigt at få patenter (Khan 2005: p. 52-56).

Den nuværende version af patentloven blev indført i 1952 og selvom software ikke er nævnt som patenterbar genstand i den, er det ud fra den at softwarepatenter er blevet patenterbart (Sterne & Bugaisky 2004: pp. 218-219). Hvordan software er kommet ind under patentbeskyttelse, vil jeg komme ind på i næste afsnit, hvor jeg vil gennemgå de tre retssager der fra 1972 til 1981 fik ændret fortolkningen af genstandsfeltet for patenter til at indeholde software.

Gottschalk v. Benson - 1972

Gottschalk v. Benson var den første software-patent-retssag der blev ført i Supreme court, og opstod da patentkontoret i USA afviste en ansøgning til en metode der ville oversætte binær kode i decimaler(BCD) til rene binære tal på en computer – en afvisning Benson appellerede af flere omgange, hvor den til sidst kom foran Supreme Court (Thomas 2008: pp. 197-198).

Tvisten mellem Gottschalk og Benson opstod, fordi Benson mente, at hans metode var patenterbar da det var en proces, og derfor opfyldte kravet i § 101:

"The legal issue before the Court in *Benson,* and in most of the subsequent cases on the patentability of computer

program-related inventions, was whether the claimed invention (in *Benson,* an algorithm for converting binary coded decimals to pure binary form) was a "process" that was patentable under the patent statute. *See* 35 U.S.C. § 101" (Samuelson 1990: p. 1028)

Det oprindelige formål med at indføre proces i stedet for art, var at opdatere lovgivning til nutidigt sprog, da art i forvejen blev tolket som en proces ved domstolen, så var det op til domstolene at tolke hvad en proces var (Sterne & Bugaisky 2004: pp. 217-218).
Men Supreme Court valgte stadig at afslå patentansøgningen da et patent af en proces skulle overholde transformationskravet ved at: ”... operate to change articles or materials to a “different state or thing.”” (Gottschalk v. Benson 1972: Section 71).
Dette gjorde Bensons patentansøgning ikke, da de digitale tal efter processen forblev digitale og derved ikke blev transformeret til en anden tilstand, hvilket betød, at metoden blev opfattet som en ren matematisk algoritme (op.cit: Section 71-72).
Men selvom det blev et afslag til Bensons software-patent slog dommeren fast, at det ikke var et afslag til den generelle patenterbarhed af software, men et afslag til abstrakte idéer, som matematiske formler og algoritmer: ”It is said that the decision precludes a patent for any program servicing a computer. We do not so hold." (Ibid: Section 71). Dette betød, at denne retssag åbnede op for muligheden for at patentere software, i stedet for at lukke spørgsmålet omkring softwares patenterbarhed, ved at afslå patentansøgningen fordi den omhandlede software.
Selvom det er dommerens job af dømme hvad der er patenterbart under patentlovgivningen, valgte dommeren ikke at tage stilling til softwares patenterbarhed, fordi det åbner op for nogle problemer som patentlovgivningen ikke lavet til at klare, da den er lavet før software var aktuelt:

"If these programs are to be patentable, considerable problems are raised which only committees of Congress can manage, for broad powers of investigation are needed, including hearings which canvass the wide variety of views which those operating in this field entertain. The technological problems tendered in the many briefs before us indicate to us that considered action by the Congress is needed." (op.cit: section. 73)

Som dommeren her er inde på, er softwarepatenter et nyt felt for patentlovgivning, og hvis man skal indføre dem, bliver man nødt til at undersøge hvilke fordele og ulemper der er ved det, da det kan få stor betydning for softwarebranchens fremtid – et syn som også bliver støttet af dommeren, i den senere retssag mellem Parker v. Flook (Parker vs. Flook 1978: Section 595). Som det førnævnt står i konstitutionen, er det derfor kongressens job at undersøge, hvordan patentlovgivningen kan skrues

sammen, sådan at den er bedst til gavn for innovationen inden for det felt og derved også til gavn for hele det amerikanske samfund.

Benson retssagen er vigtig fordi den åbner op for at software kan patenteres, samtidig med at den sætter fokus på, at patentlovgivningen ikke bygger på konstitutionelle rettigheder, men er et produkt at den tid den er lavet i, og derfor skal tilpasses til de nye felter og forhold der opstår på erhvervsmarkedet.

Parker v. Flook - 1978

Mens at Benson-retssagen omhandlede et patent af et computerprogram, havde patentet i Parker v. Flook flere dele, da det var en fysisk proces der blev styret af et computerprogram: ”*Flook* differs from *Benson* primarily because *Benson* involved a computer program per se while Flook's claim was for a process that includes a computer program" (Haughey 1978-1979: p.1637).

Flook prøvede at patentere et alarmsystem der automatisk omregner grænseværdier i en katalysator-proces, og starter alarmen hvis nogle af de forskellige grænseværdier for processen bliver oversteget (Parker v. Flook 1978: section 584).

Det store spørgsmål her er, hvorvidt en proces, der ellers ville være patenterbar, ville blive upatenterbar når den indeholder et softwareprogram - da det derved ville fjerne alle muligheder for at få software ind under den daværende patentlov.

Dommeren brugte retssagen mellem O'Reily og Morse som argument for, at dette kunne være muligt. Morse havde ansøgt et patent på at implementere princippерne for elektromagnetisme til at lave telegrafen: "The Court in *Morse* went on to indicate that the flaw in Morse's claim was that it was too broad; if it had been restricted to the specific method Morse had used, it would have been allowed" **(**Haughey 1978-1979: p. 1639**)**.

Flooks patentansøgning havde derfor en mulighed for at blive godkendt, da det var en ny implementering af en i forvejen kendt formel.

Men patentet blev afslået, fordi den ikke indeholdt noget innovativt i selve implementeringen af formlen – selvom den var ny (Parker v. Flook 1978: Section 594).

Derfor ender dommeren ligesom, ved Benson, med ikke at tage stilling til softwares patenterbarhed, ved at afslå den på andre grunde, da han ligesom dommeren hos Benson mener, at sådan en beslutning ikke skal tages af domstolen (Parker v. Flook 1978: Section 595).

Men samtidig åbner Flook retssagen muligheden op for, at patentere software, hvis man opfinder en innovativ måde at implementere den, samtidig med at man overholder de normale kvalifikationer

for en patentansøgning.

Parker v. Flook flytter derved spørgsmålet om softwares patenterbarhed væk fra de grundlæggende algoritmer og koder, hen til et spørgsmål om det innovative i implementeringen af disse.

Diamond v. Diehr - 1981

I retssagen med Diamond v. Diehr blev det så bekræftet, at en given proces der ellers ville være patenterbar, stadig var patenterbar selvom at den indeholdt software – da det var det første softwarepatent der blev godkendt (Thomas 2008: pp. 200-201). Diehr søgte et patent hvor et computerprogram styrer en fysisk proces, som kan forme rå syntetisk gummi til vilkårlige elementer – en proces der kræver at faktorer som temperaturforhold er helt præcise, hvilket programmet er med til at sørge for (Diamond v. Diehr 1981: Section 175).

Diehr fik godkendt patentet ved Supreme Court, fordi det, i forhold til Benson, bl.a. overholdt kravet om transformation af et stof, da den syntetiske gummi bliver transformeret til en anden tilstand pga. denne opfindelse (Ibid). Samtidig faldt patentet ikke for kravet om nyhedsværdi som hos Flook, da dommeren i Diamond v. Diehr så kravet om nyhedsværdi som et hele, i stedet for at se om de enkelte dele af processen overholdte dette krav (Thomas 2008: p. 200).

Året inden Diehr havde genstandsfeltet allerede skubbet sig gevaldigt, da dommeren i retssagen mellem Diamond v. Chakrabarty definerede det til at indbefatte: ”anything under the sun that is made by man." (Diamond v. Chakrabarty 1980: Section 309). Selvom der senere i domsafsigelsen blev uddybet, at dette ikke var ment som en direkte udvidelse af genstandsfeltet, men mere et forsøg på at fjerne arbitrære forhindringer for at udvide det – som det senere blev med Diamond v. Diehr (Ibid; Thomas 2008: pp. 200-202).

Forhindringen for indførelsen af softwarepatenter i kraft af §101 blev i perioden efter Diamond v. Diehr, endnu mindre.

Gennem retssagerne med State Street Bank i 1998 og AT&T i 1999, der begge omhandlede rene erhvervsmetoder, blev kravet om transformation af et stof lempet, da transformationen blev tolket til at være en af de mange mulige måder et patent kunne blive godkendt på – i stedet for at alle patenter skulle overholde det (Thomas 2008: pp. 206-207). Samtidig blev genstandsfeltet også udvidet til at indeholde erhvervsmetoder, der, ligesom software, tidligere blev tolket til at være abstrakte ideer - hvilket banede vejen for rene softwarepatenter.

Diamond v. Diehr viser at software ikke blev patenterbart pga politisk intervention, i kraft af ændringer i patentlovgivningen. Software blev patenterbart fordi Supreme Court ændrede holdning

til, hvordan patentlovgivningen skulle tolkes, hvilket betyder at software blev patenterbart, uden at have et samfundsøkonomisk formål – som ellers er det formål der i konstitutionen legitimerer patentlovgivningen.

Apple Inc v. Samsung Electronics Co., Ltd

Apple v. Samsung er en række af retssager, som i 2012 omhandlede over 50 retssager i 10 lande, hvoraf søgsmålene kommer fra begge parter (Mueller 2012). Den første retssag startede i 2011, da elektronikgiganten Apple sagsøgte Samsung i USA for at kopiere Apples Iphone, og derved bryde mange af Apples patenter (Lee 2012: pp. 119-120).

Patent krigen mellem Apple og Samsung skiller sig ud fra andre tidligere patent-krige, fordi den består af så mange forskellige patenter, samtidig med, at det er begge parter der sagsøger hinanden – da Samsung få måneder efter de blev sagsøgt, sagsøgte Apple for brud på nogle af deres patenter (Ibid). Samtidig har den også fået stor opmærksomhed, da det er første gang at to firmaer sagsøger hinanden over hele jordkloden, med det formål at fjerne konkurrentens produkter fra markedet (Lee 2012: pp. 117)

I resten af opgaven vil jeg med udgangspunkt i disse retssager, undersøge de forskellige problemer softwarepatenter skaber, og hvilke muligheder er der er for at løse dem.

Kvantitativt frem for Kvalitativt

Siden det blev muligt at patentere software, er antallet af patentansøgninger og antallet af retssager om patenter steget voldsomt, hvilket ifølge Jaffe & Lerner er tegn på en stor tvivl om hvad der kan patenteres og hvor meget disse patenter beskytter (Jaffe & Lerner 2006: pp. 3-4).

Målet for patentansøgningerne bliver derfor kvantitative frem for kvalitative, da man ikke kan vide sig sikker på at det givne patent beskytter ens produkt, og derfor sikrer en virksomhed sig bedst ved at søge mange patenter (Jaffe & Lerner 2006: pp. 3-4).

Ifølge Robert E. Thomas kommer denne tvivl bl.a. fordi softwarepatent-retssager er væsentlig mere komplekse end tidligere typer af patent-retssager, hvilket betyder at det er sværere for dommeren og de 2 parter i retssagen at blive enige om hvad en given krænkelse af et patent handler om og om det givne patent er gyldigt (Thomas 2008: p. 228). Det ligger altså immanent i det tekniske ved software, at der er en større usikkerhed om softwarepatenters gyldighed i en retssag**.**

Når man derved aldrig kan vide sig sikker om ens patent beskytter, eller om andre har patenter der kan bruges mod en, skaber det ikke et incitament for de små, eller store virksomheder til at være

innovative, da en eventuel retssag mod dem vil være for dyr en chance at tage, når usikkerheden om dens resultat vil være for stor (Jaffe & Lerner 2006: pp. 3-4).

Det store antal af patenter gør det også svært for nye patentansøgere at finde ud af, om deres opfindelser bryder nogle af de tidligere udstedte patenter, da det kræver store ressourcer at undersøge samtlige software patenter – hvilket igen skaber en forhindring for at skabe innovativt software (Lichtman & Lemley 2007: pp. 46-48).

Men usikkerheden omkring softwarepatenters gyldighed skyldes ikke kun at software-patent-retssager er komplekse, men er ifølge Jaffe & Lerner også et resultat af dårlige undersøgelser af patenters gyldighed, fra PTOs side – hvilket jeg vil kommer ind på næste afsnit (Jaffe & Lerner 2006: pp. 5).

Godkendelsen af patenter

Patenterne som Samsung blev sagsøgt for at bryde, bestod hovedsagligt af designpatenter, men samtidig også softwarepatenter, som patentet på at scroll'e rundt på en trykfølsom skærm (Apple Inc v. Samsung Electronics Co., LTD 2013: pp. 5-8; U.S Patent 7,844,915 2010)[2].

Jeg undrer mig over at et patent som scroll-patentet overhovedet kan gå igennem patentansøgningen, da der er lavet utallige science fiction film med touch-skærme, hvor personer, via hænderne, scroll'er på skærmen (f. eks. Minority Report fra 2002) - hvilket burde betyde at patentet ville falde for nyhedskriteriet, da ideen i forvejen er tænkt og publiceret i det offentlige rum.

Men som Jaffe & Lerner skriver, har PTO[3] svært ved at opretholde de krav der er fremsat for godkendelsen af patentansøgninger:

”Again, the PTO appears not to have done a good job at making sure that applications for software patents are tested against this non-published prior art. The result has been a deluge of patents granted on software concepts that appear not to be new." (Jaffe & Lerner 2006: p. 27)

Dette skaber et problem, da det betyder at mange af patenter der blive brugt i retssager, ikke skulle være udstedt til at starte med (Ibid).

Ifølge Jaffe & Lerner sker dette bl.a. fordi softwarepatenter er så nye, at man kun har een patentdatabase for softwarepatenter der går tilbage til 1981, og derfor skal man finde tidligere

2 I resten af opgaven vil jeg referere til dette patent som scroll-patentet.

3 PTO = United States Patent and Trademark Office

publiceret software i andre databaser (Ibid). Samtidig er antallet af patentansøgninger steget i løbet af de sidste mange år, hvilket betyder at PTO har færre ressourcer til at gennemgå de forskellige patentansøgninger og derfor er der en større sandsynlighed for fejl i processen (Jaffe & Lerner 2006: p. 3, 6).

Men det største problem opstår ifølge Jaffe & Lerner, fordi PTO ikke længere overholder kravet om en detaljeret beskrivelse af implementeringen af patentet, og derfor er det de færreste softwarepatent ansøgninger der skriver sådan en, da det ikke er nødvendigt for at få godkendt patentet (Ibid).

Når patentansøgerne ikke behøver at vise, at de kan få implementeret deres patent, åbner det op for at firmaerne kan føre rovdrift på software-patenter, hvor de kan ansøge om patenter de ved de ikke kan bruge kommercielt, og derefter vente til et andet firma beslutter sig for at lave et produkt, sådan at de kan sagsøge dem for brud på deres patenter. Dette ville ikke være muligt hvis der var et krav på en detaljeret beskrivelse af implementeringen af patentet:

”The result has been a flood of patent applications on myriad diverse software ideas; in principle the recipient of such a patent then has the right to exclude others from implementing the covered software idea, despite the fact that they have never implemented, or even described implementing, the idea themselves" (Jaffe & Lerner 2006: p. 27)

Disse virksomheder der ikke selv implementerer deres patenter i brugbare opfindelser, kaldes Patent-trolls (Rantanen 2006: p. 160). Det er firmaer der ikke selv prøver på at kommercialiserer deres produkter, men kun tjener penge på at sagsøge andre for at bryde deres patenter, vel og mærke flere år efter at et givent produkt har fået succes, sådan at erstatningen er stor nok til at dække udgifterne for en eventuel retssag (Ibid).

For Jaffe & Lerner er løsningen ikke at fjerne muligheden for at patentere software, men at stramme overholdelsen af de krav der i forvejen er, samt genindføre kravet om en detaljeret beskrivelse af implementeringen af patentet (Jaffe & Lerner 2006: pp. 31-32). Et krav der vil gøre det langt sværere at få godkendt et patent, hvilket sandsynligvis vil sænke antallet af patentansøgere.

Med færre antal ansøgninger vil det amerikanske patentkontor have muligheden for at bruge flere ressourcer på de ansøgninger, sådan at de ansøgninger der bliver godkendt, er grundigt testet til at overholde de krav som PTO sætter for patenter, og derved innovative (Jaffe & Lerner 2006: pp. 14-15).

Som det er nu, bliver et patent kun betegnet som værdifuldt hvis det er blevet testet i en retssag, da alle patenter af økonomisk signifikans ville blive testet i en retssag (Allison, et al. 2003: p. 55).

Ved at gøre det sværere at få godkendt et patent, forhøjer man ifølge Jaffe & Lerner, værdien af patenterne der ikke kommer igennem en retssag, selvom man selvfølgelig ikke kan undgå retssager, da der altid vil komme til at ske fejl i PTO (Jaffe & Lerner 2006: pp. 31-32).

Det vage ved patenter

Det uforudsigelige i en retssags udfald er ikke kun et resultat af at softwarepatenter er komplekse af natur, men som Ismael Arinas skriver i sin artikel *How Vague can a patent be?*, er det et bevidst middel for patent-ansøgere, at sørge for at patentet er så vagt som muligt – og derved bredtfavnende: ”Vagueness is seen often as a problem when it comes to interpret legal language ... But if we approach vagueness from the point of view of the drafter of legal documents, then it becomes a useful and desirable feature." (Arinas 2012: p. 55).

Det at patentet er vagt, vil i Arinas forstand ikke betyde at det ikke er præcist, men mere at selve måden hvorpå patentet bliver fuldført ikke er determineret – sådan at man har mulighed for at justere på de forskellige dele for at få opfindelsen til at fungere (Arinas 2012: pp. 57-58).

Samtidig bruges vage formuleringer også til at sikre sig imod fremtidige problemer og utilsigtede søgsmål fra konkurrenter (Arinas 2012: p. 59-60). Derfor er det ifølge Arinas logisk for en given patentansøger, at gøre patentet så vagt som muligt, samtidig med at det kan blive godkendt.

Arinas pointerer også, at det samtidig er logisk at gøre patenter vage, fordi de love som patenterne skal indordne sig under, selv er vage og derfor er man nødt til at lave patenterne vage for at de kan være brugbare (Ibid).

Når et firma forsøger at lave deres patenter så vage som muligt, giver det derved et ekstra lag af kompleksitet ved retssager med softwarepatenter, da et vagt patent vil gøre det sværere at definere hvad patentet beskytter og hvad det ikke beskytter.

I Apple v. Samsung ligger det vage i patentet allerede i det faktum, at der ikke står skrevet i f. eks. scroll-patentet, hvordan de vil implementere det. Dette giver dem derfor muligheden for at fremtidssikre patentet mod uforudsete implementeringer af det (Ibid).

Et vagt patent har samtidig den fordel, at det skjuler for konkurrenterne, de tekniske detaljer for implementeringen af patentet.

Hvorfor softwarepatenter?

Mens en af mulighederne for at løse problemerne med softwarepatenter er at stramme grebet om patentansøgninger, er en anden løsning helt at fjerne muligheden for at få softwarepatenter.

Nogle af modstanderne mod softwarepatenter, argumenterer for at softwarebranchen i 1970'erne ikke led under manglende patenter, men faktisk blomstrede lige så meget som den har gjort efter indførslen af software patenter (Jaffe & Lerner 2006: p. 28).

I en anden kritik af softwarepatenter, argumenterer Robert E. Thomas mod at have patentbeskyttelse af software, da det mindsker innovation, og derved mindsker den velfærd som staten ville få ud af en forhøjet innovation (Thomas 2008: pp. 223-224).

Han støtter på de fleste punkter Burk og Lemleys arbejde med innovation og patentlovgivning, hvor de mener at patentlovgivning skal tilpasses de forskellige brancher, da incitamenterne for at innovere, er forskellige i f. eks. medicinalbranchen og softwarebranchen (Thomas 2008: pp. 214-215).

Et eksempel herpå ville være R&D[4]-omkostningerne for et medicinalfirma, som er højere end de er for et softwarefirma, samtidig med at det er nemt for en konkurrent til medicinalfirmaet, at kopiere produktet når det udkommer. Dette betyder, at medicinalfirmaer har behov for patentbeskyttelse, da det ellers ikke ville kunne betale sig for et medicinalfirma at innovere (Thomas 2008: pp. 215-216).

Burk og Lemley deler innovation op i fire forskellige typer, hvoraf den moderne softwarebranche ifølge dem, passer til cumulative-innovation hvor: ”inventors progress in small, incremental steps building off the work of their predecessors.' In such industries, optimal innovation occurs when patent rights are narrow, there by giving inventors incentives to create without blocking the path of subsequent creators" (Thomas 2008: p. 213).

Patentsystemet som Burk og Lemley derfor vil indføre for at øge innovationen i software-branchen, minder meget om den som Jaffe og Lerner havde, hvor software-patenterne skal indsnævres, sådan at andre har mulighed for at bygge videre på ens patenter – uden der bliver skabt monopol.

Mens Thomas er enig med Burk og Lemley i nogle af deres definitioner af innovation, mener han imidlertid også at de begår en fejl i deres arbejde med softwarebranchen, fordi deres udgangspunkt er at skabe det størst mulige incitament for innovation. I stedet mener Thomas at man skal se på hvad der på markedsbasis bringer størst innovation, hvilket, ifølge ham, vil ske hvis man helt fjerner muligheden for at patentere software (Thomas 2008: pp. 214-215).

Det vil det bl.a. pga. de lave R&D udgifter der er for at udvikle patenterbart software, hvilket

4 Research and Development

betyder, at firmaer, der satser meget på innovation, ikke har de samme problemer med at tjene omkostningerne hjem som medicinalfirmaer ville have, hvis der ikke var patentsikkerhed (Thomas 2008: p. 218).

Men som jeg her kommer ind på, er der et problem ved Thomas' grundlag, da der er utallige eksempler på dyrt innovativt software. Et eksempel herpå er Apples første iPhone, der skulle have kostet over 150 millioner af dollars at udvikle, hvilket kunne have betydet en konkurs hvis telefonen ikke var blevet en succes – et andet eksempel er det danske NEM ID der samlet set har kostet 871 millioner (Vogelstein 2013; Hansen 2010). Som Thomas selv er inde på, kræver det også store summer penge at komme ind på et software marked, da et givent stykke software får en større værdi jo flere på markedet der bruger det – da de forskellige kunders produkter derfor er kompatible med hinanden (Thomas 2008: pp. 221-222)

Men dette er Thomas bevidst om, da han ikke mener at disse udgifter ligger i udviklingen af ideer og patenter, men i implementeringen og kommercialiseringen af dem - da man ikke behøver at implementere ideen for at få den patenteret (Thomas 2008: pp. 219-220).

Når patenteringen af software er billig og implementeringen er dyr, kan man derfor ligeså godt fjerne patenterne, da det stadig er de færreste der alligevel ville have råd til at kommercialisere deres produkter uden patentsikkerhed.

Dette vil frembringe større innovation da de små firmaer kan bruge den viden de store firmaer har, og få den implementeret i nogle produkter, ved at sælge deres ideer videre til de store selskaber – den samme struktur som der lige nu er i Silicon Valley, hvor 'start ups' bliver solgt for store millionbeløb, fordi de har udviklet noget software som en af de store firmaer vil have implementeret i deres produkt (Bright 2011).

Det at det er dyrt og svært at implementere software i produkter, er samtidig også styrken for softwarebranchen, da der er en naturlig fortjeneste for at investere i innovation, hvilket Apple 's iPhone er et eksempel på.

Apple fik og har kæmpe succes med sin iPhone, hvor den tog store markedsandele, da det tog tid for deres konkurrenter at producere en smartphone der fungerede lige så godt som den – en gevinst som automatisk kommer inden for IT branchen, da det tager tid for selskaber at udvikle programmer, samtidig med, at det ifølge Thomas er svært at få flyttet kunder fra et produkt til et andet, når man først har vænnet sig til at bruge det (Canalys 2009; Thomas 2008: pp. 221-222).

Samtidig viser Apple i retssagen mod Samsung netop den ageren, som Thomas tillægger de store virksomheders brug af software patenter - de prøver at stoppe konkurrenterne gennem aggressiv

brug af sine patenter. En ageren der resulterer i mindre innovation, hvor det modsatte ellers er formålet med patenterne.

Selv en snæver definition af softwarepatenter vil ifølge Thomas mindske incitamentet for innovation for de dominerende firmaer på markedet, da de i forvejen tjener mange penge på de patenter de har (Thomas 2008: pp. 217-218). Derfor vil de ikke have nogen grund til at investere i R&D til nye produkter, da det ville udkonkurrere deres eget produkt, og derfor ikke resultere i en større omsætning - ligesom de vil forsøge at bremse andre med at udvikle nye konkurrerende produkter. Løsningen for Thomas er ikke direkte at forbyde softwarepatenter i patentlovgivning, med i stedet at indføre og overholde transformations-testen for godkendelsen af patenter, da det vil fjerne muligheden for at kunne patentere software (Thomas 2008: p. 241).

Konklusion

Målet for indførelsen af patenter i USA var at fremme videnskaben og innovationen, til fordel for velfærden i det amerikanske samfund. Det var og er et middel for staten til at skabe størst mulig incitament for alle, til at skabe innovation inden for deres branche. Derfor undrer det mig også at staten ikke har udnyttet dette redskab, til at skabe regler der kunne fremme innovationen, inden for de nye videnskabelige og tekniske felter.

I stedet har patentlovgivningen levet et liv for sig selv, hvor den er blevet formet af retssager i Supreme Court. Dette har bl.a. ført til en indførelse af softwarepatenter som patenterbare objekter, da det i Diamond v. Diehr ikke kunne forsvares at konsekvent holde nye tekniske felter ude for patentlovgivningen, da patentlovgivningen var universel og lavet til at favne alt hvad der er menneskeskabt.

Men når patentlovgivningen favner så bredt, uden at have specifikke definitioner for hvordan de forskellige typer patenter skal behandles, skaber det usikkerhed omkring patenternes gyldighed.

Dette sker bl.a. fordi ansøgnings processen for patenter er blevet dårligere, da mange af kravene for godkendelse af patenter er lempet eller væk, samtidig med at de krav der er der, ikke bliver overholdt.

Dette har ført til en mere aggressiv patentstrategi for elektronikfirmaer, hvor Apple v. Samsung er det nyeste eksempel på virksomheder der bruger patenter som et våben mod konkurrenter, hvor ingen kan vide sig sikre, da resultaterne af retssagerne er umulige at forudsige – en strategi der gør det usikkert for firmaer at satse på innovation, da de hverken kan være sikre på at deres patenter beskytter deres opfindelser, og om deres opfindelser bryder nogle andres patenter.

Siden målet med patentsystemet er at fremme innovation, mener jeg derfor at det amerikanske patentsystem ikke længere opfylder det mål indenfor software-branchen, da incitamenterne for at innovere ikke bliver højere pga den nuværende patentlovgivningen, men nærmere lavere, da usikkerheden om patenternes gyldighed er til fordel for de store virksomheder der har råd til at føre retssager.

I en videre undersøgelse af software patenter, ville det være oplagt at undersøge hvilken påvirkning globaliseringen og internettets udbredelse har på patentlovgivningens gyldighed, da patentlovgivningen i alle lande er nationalt funderet og derfor bliver nød til at tilpasse sig de nye forhold for at blive international.

Jeg har i idéhistorisk forstand, prøvet at afselvfølgeliggøre software som et patenterbart objekt, og derved undersøge hvilke ideer der lå til grund for at gøre det patenterbart. Men der var ikke nogen bagvedliggende ide med at godkende softwarepatenter, og det er pga disse manglende ideer, at software patent lovgivningen har svært ved at udfylde sit formål, om at fremme innovationen indenfor software-branchen. Derfor skal den amerikanske kongres finde ud af hvilke fordele og ulemper der er ved softwarepatenter, og derefter tage den beslutningen der fremmer innovationen, og incitamentet for innovation, mest.

Litteraturliste

Allison, John R, et al. (2003): *Valuable Patents.* Boalt Working Papers in Public Law, Boalt Hall, UC Berkeley.

Apple Inc v. Samsung Electronics Co., Ltd (2013): United States Court of Appeals for the Federal Circuit. Appel nummer 13-1129, tilgængelig på domstolens hjemmeside: http://www.cafc.uscourts.gov/images/stories/opinions-orders/13-1129.Opinion.11-14-2013.1.PDF (12/12 - 2013)

Arinas, Ismael (2012): *How Vague Can Your Patent Be? - Vagueness Strategies in U.S. Patents.* HERMES - Journal of Language and Communication in Business, No. 48, pp. 55-74.

Bright, Peter (2011): *Microsoft Buys Skype for $8.5 Billion. Why, Exactly?*, tilgængelig på Wireds online magasin: http://www.wired.com/business/2011/05/microsoft-buys-skype-2/ (12/12 - 2013)

Canalys (2009): *Smart phones defy slowdown - Nokia retains lead, with Apple and RIM rising fast, but Microsoft loses ground*, tilgængelig på Canalys egen hjemmeside: http://canalys.com/newsroom/smart-phones-defy-slowdown

Cohen, Julie E & Lemley, Mark A. (2001): *Patent Scope and Innovation in the Software Industry.* California law Review, Vol 89, No. 1, pp. 1-57.

Constitution of the United States (1787): Philadelphia, de nationale amerikanske arkiver, http://www.archives.gov/exhibits/charters/constitution_transcript.html (7/12 - 2013)

Diamond v. Diehr (1981): U.S. Supreme Court. Page 450, United States Reports 175, tilgængelig på: http://supreme.justia.com/cases/federal/us/450/175/ (7/12 - 2013)

Diamond v. Chakrabarty (1980): U.S. Supreme Court. Page 447, United States Reports 303, tilgængelig på: http://supreme.justia.com/cases/federal/us/447/303/case.html (7/12 - 2013)

Forbes (2013): Forbes Global 2000, Forbes årlige opdaterede liste over verdens 2000 største virksomheder, tilgængelig på: http://www.forbes.com/global2000/list/#page:1_sort:0_direction:asc_search:_filter:Software%20%26%20Programming_filter:All%20countries_filter:All%20states (10/12 - 2013)

Gottschalk v. Benson (1972): U.S. Supreme Court. Page 409, United States Reports 63, tilgængelig på: http://supreme.justia.com/cases/federal/us/409/63/case.html#F5 (7/12 - 2013)

Hansen, Kristian (2010): *Så mange millioner kroner koster NemID-projektet*, tilgængelig på: http://www.computerworld.dk/art/112204/saa-mange-millioner-kroner-koster-nemid-projektet (7/12 - 2013)

Haughey, Poul (1978-1979): *Parker v. Flook and Computer Program Patents.* Hastings Law Journal, Vol 30, pp. 1627-1644.

Jaffe, Adam & Lerner, Josh (2006): *Innovation and Its Discontents.* Capitalism and Society, Vol 1, Iss 3, Article 3.

Khan, B. Zorina (2005)*: The Democratizaion of Invention – Patents and Copyrights in American Economic Development, 1790-1920.* New York, Cambridge University Press.

Lee, Jaemin (2012): A Clash between IT Giants and the Changing Face of International Law: The Samsung vs. Apple Litigation and Its Jurisdictional Implications. Journal of East Asia & International Law, Vol 5, No 1, pp. 117-142.

Lichtman, Doug & Lemley, Mark A. (2007): *Rethinking Patent Law's Presumption of Validity.* Stanford Law Review, Vol. 60, Iss 1, pp. 45-72

Mueller, Florian (2012): *List of 50+ Apple-Samsung lawsuits in 10 countries,*
tilgængelig på: http://www.fosspatents.com/2012/04/list-of-50-apple-samsung-lawsuits-in-10.html (7/12 - 2013)

Parker v. Flook (1978): U.S. Supreme Court. Page 437, United States Reports 584, tilgængelig på: http://supreme.justia.com/cases/federal/us/437/584/case.html (7/12 - 2013)

Patent Act of 1793 (1793): Washington D.C, University of New Hampshire's net bibliotek over intellektuel ejendom, http://ipmall.info/hosted_resources/lipa/patents/Patent_Act_of_1793.pdf (7/12 - 2013)

Pwc (2012): *Patent Litigation Study 1995-2011*, tilgængelig på: http://www.pwc.com/en_US/us/forensic-services/publications/assets/2012-patent-litigation-study.pdf (12/12 - 2013)

Rantanen, Jason (2006): *Slaying the Troll: Litigation as an Effective Strategy against Patent Threats.* Santa Clara Computer & High Technology Law Journal, Vol 23, Iss 1, Article 5.

Samuelson, Pamela (1990): *Benson Revisited: The Case Against Patent Protection For Algorithms and Other Computer Program-Related Inventions.* Emory Law Journal, Vol 39, pp. 1025-1154.

Sterne, Robert Greene & Bugaisky, Lawrence B. (2004): *The expansion of Statutory Subject Matter Under The 1952 Patent Act.* Akron Law Journal, Vol 37, pp. 217-228.

Title 35 of the United States Code (1952) Washington D.C, hentet fra Cornell Law School database over Amerikanske love, http://www.law.cornell.edu/uscode/text/35 (7/12 – 2013).

Thomas, Robert E (2008): *Debugging Software Patents: Increasing Innovation and Reducing Uncertainty in the Judicial Reform of Software Patent Law.* Santa Clara Computer & High Technology Law Journal, Vol 25, Iss 1, Article 7.

U.S Patent 7,844,915 (2010) Tilgængelig på PTO's internet database: http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.htm&r=1&f=G&l=50&s1=7,844,915.PN.&OS=PN/7,844,915&RS=PN/7,844,915

Vogelstein, Fred (2013): *And Then Steve Said, 'Let There Be an iPhone'*, tilgængelig på: http://mobile.nytimes.com/2013/10/06/magazine/and-then-steve-said-let-there-be-an-iphone.html (7/12 - 2013)