Mag. iur. Dr. techn. Michael Sonntag

Software Patents

Inventors assistant
Johannes Kepler University Linz

E-Mail: sonntag@fim.uni-linz.ac.at
Telefon: +43(732)2468-**9330**
http://www.fim.uni-linz.ac.at/staff/sonntag.htm

Agenda

- Software Patents
  - → What is special about software patents?
  - → What does "as such" mean?
- Core theory vs. theory of holistic consideration
- The (failed) EU directive on software patents
- Exemplary patents
- Software: Patents vs. copyright
- Software patents in the USA and Japan

What are software patents

- Software patent: A patent where the invention consists (also) of software
  - → Depends on the definition of software!
    - » Instructions for automatic execution by a computer
      - – Could theoretically also be an analogue computer!
- In theory, there are no software patents at all in Europe
  - → "The EPO did not issue any software patents"
  - → But why are there then about 30.000 patents regarding SW?
    - » But why then is their no infringement litigation?
- Why are there currently no real problem for companies regarding defending against SW patents?
  - → And what problems/disadvantages exist for companies because they cannot obtain SW patents?
- Who are the drivers behind the patentability of SW?

What is special about software regarding patents?

- May the software be an "accessory", which is protected alongside a "normal" invention?
  - → A machine which also contains some kind of computer
- What about "pure" software inventions?
  - → The new idea is only part of the software, but not the hardware
- Can they protect unpatentable things implemented in SW?
  - → Games are not patentable. What about computer games?
- Separation of the problem from the implementation
  - → No program code allowed → Where's the difference to a problem statement?
- What's the difference between an algorithm and a mathematical method?
  - → But: Both similarly only excluded "as such"!

Software vs. physical engines

- "SW is different: Machines vibrate, are inexactly produced, might have resonance, … - Programs are mathematics"
  - → This is certainly true for small programs
  - → But large programs are very prone to resonance (livelocks), vibration (race conditions), inexact (bugs), etc!
- "Software is only a plan – You cannot patent those"
  - → A procedure for distilling some drug is also only a "plan"!
    - » What to do in which order and with certain parameters
- "Machines are also built of many parts"
  - → In general, very few "machines" are built from 100.000 parts – But 100.000 LoC are not that uncommon or large!
- "A builder of cars must also consider many patents"
  - → But much less than a sizeable program!
    - » No "subclass" for SW patents or a subdivision

The legal situation

- Both the EPA and the AT/DE patent laws forbid patenting software "as such"
  - → PCT examination authorities are not required to examine computer programs if not equipped to do so (Rules 39, 67)
    - » "Science and mathematical theories" → No requirement at all
- Courts try to find a meaning for "as such" and reach widely differing results
  - → EPO: Very wide
    - » If it contains something "technical", it is patentable
  - → DE: In general possible under certain restrictions
    - » Previously rather few, then many, now again more restrictive
  - → AT: Similar to DE
    - » Very few decisions (or information) available
- USA: Software can be patented without problem
  - → Currently strong push to reduce the scope!

"As such" – A literal approach

- What does "as such" mean when interpreting the words?
  - → Only look at the text: What is the smallest and what the widest possible meaning?
- Possible meanings:
  - → Its essence, the main characteristics only
    - » Only program code is excluded, everything else is possible
  - → Without contemplating any specific usage
    - » Every program with a specific purpose could be patented
      - – What is an example of a program without any purpose???
  - → Without any restrictions
    - » Nothing including a program could be patented at all
  - → Separately from the machine executing it
    - » No patent on program, but on "program running on a computer"
- Very difficult to give this short fragment a consistent meaning, as it is used in a wide variety of situations!

"As such" – A literal approach

- Commentary (Schulte):
  "A patentable invention may be based on a discovery, aim at an aesthetic effect or employ a computer program." [translated form German]
  - → Note: Patent is not on discovery, effect or program!
    - » E.g., new plant species discovered → Drug patent based on it
    - » E.g., new method for painting a pattern simulating marble
    - » E.g., new chemical process controlled by a computer
- See also the Berne convention (copyright) Art 2 (5):
  "Collections of … works … which, by reason of the selection and arrangement of their contents, constitute intellectual creations shall be protected as such, …"
  - → General assumption: A collection may be copyrighted, but remains independent from its elements and their protection
    - » Applying this to patents: No patents on software (i.e. the program itself), but what a program consists of/runs on/achieves/is used for might be patented

"As such" – A teleologic approach

- What are the aims of this restriction?
  - → What should be achieved or prevented through it?
- Obviously there is to be made a distinction:
  - → Some programs may be patented, and some not
    - » Because there are general exemptions, and those "as such"
- Regrettably, when this text was passed, there was a general agreement, that no definition is possible
  - → "It will remain for the courts to provide guidelines"
    - » This is problematic from a basic view: The (continental!) law should define what is allowed or forbidden, and not leave it open to the courts to decide this!
  - → Another reason was also the very fast speed of development
    - » What exactly a computer can/cannot do was not apparent

A comparison to other exclusions with the "as such" limitation

- Mathematical methods:
  - → Faster calculation of square roots is not patentable (decision)
    - » This is "abstract", i.e. "pure" mathematics → mathem. "as such"
  - → Faster compilation of programs could not be patented
    - » Note: Faster execution of programs might be (and was!)
- Aesthetical creations:
  - → Nothing patentable producing specific aesthetics
    - » But how to produce them is patentable
  - → Methods/machines creating programs could be patentable
    - » But in general these are either humans or programs …
      - – Methods: Business methods, rules for mental activities, …
- Presentation of information:
  - → Forms cannot be patented
  - → User interfaces could not be patented
    - » But see SOHEI decision!

The core theory (1)

- The essence of the invention must be checked
  - → Only if the essence is technical and inventive, i.e. fulfils all requirements, the patents can be granted
  - → The "new" element must also be the "inventive" (and …) one!
    - » But it may be realized trough a computer
- If it is a mental act, it will not become technical through execution on a computer
  - → "Adding a computer" does not make anything technical
    - » Executing business methods on a computer → Still not technical!
- Another example: Improved washing machine
  - → Better washing through controlled dispensing of detergents
  - → The dispensing is controlled by a computer
  - → But new (and inventive, …) is when/how to dispense the detergent, not how to implement it
  - → Only this method is patented, not the software implementing it

The core theory (2)

- A solution must provide a new teaching on the use of controllable forces of nature without human decisions
  - → Technical solution
- Basic decision: "Red dove":
  - → A specific method for breeding animals is patentable and technical, as it can be controlled and employs forces of nature
- Typical SW example: Anti blocking system
- Examples for excluded methods:
  - → Sorting
  - → Minimizing flight costs through fuel consumption regulation
    - » The software does automatically what otherwise the pilot would (and could) have done ("high-level" fuel regulation)
      - – A kind of "organizational rule", i.e. an economic problem solved by a (standard) computer
      - – Flying like this has no technical effect, only a monetary one

The theory of holistic interpretation ("Technical contribution")

- The invention must be examined as a whole
  - → There must be something technical, something inventive, something new, …
    - » But these need not be the same part!
    - » I.e., the software is new, it runs on a (technical) computer, and the display of the result is inventive
- Typical example: Speech analysis
  - → "A computer (i.e. hardware) characterized through a program"
    - » If a program is new and inventive, it can be patented
      - – This would also include business methods!
- Later reduced: "Solely" adding a computer is insufficient
  - → Some "technical problem" is required
    - » Solving an economic problem with the computer → Unpatentable
  - → This leads to the Vicom decision – core and holistic mixed
    - » A program is patentable, if it involves a technical consideration

Technicality / Technical considerations (1)

- Requirement: Solving a technical problem
  - → How this is achieved is unimportant
  - → Conclusion: If the computer could theoretically be replaced by a machine (but not a human who must decide something!), then it can be patented
  - → SW solving a techn. problem would then be no SW "as such"
- Result: Every program solving the same problem in the same way requires a license
  - → Note: The same program solving a different problem is not affected, neither is solving the same problem in a different way, even through a program, affected
- Essence: A process for doing something in a certain way is patented, which is just "accidentally" performed by (or through) a computer

Technicality (2)

- Technicality in this definition does not require anything "physical" at all!
  - → No "forces of nature" (But: Electrons moving through silicon?)
- The "technicality" need not be present in the "solution"!
  - → It might also be only the problem, which is technical!
- Potential problem: What if the problem can only be solved in a single way or only with a program (but not mechanically)?
  - → See the "merger" doctrine in copyright law!
- Examples for borderline problems:
  - → "Performing calculations more efficiently"
    - » Requires less power and time in a computer: Technical problem
    - » Solved through better memory layout: Non-technical solution
  - → Vicom: New mathematical operations on digital images
    - » Filtering an image: Technical problem
    - » Matrix operation: Non-technical solution

Comparing the theories

- The core theory is more restrictive
  - → Fewer inventions will match this criteria
  - → It is especially difficult for software to match all
    - » Generally, software is only an "accessory"
    - » Software cannot contribute to "new" and "inventive"
    - » "A new and inventive physical process" + computer
- Theory of holistic interpretation
  - → Very few inventions will not match these criteria!
  - → In its pure form it is not accepted
    - » Those few decisions are regarded as erroneous by most
    - » Requires a technical consideration
- Common ground:
  - → An invention must be "technical"
  - → Solution vs. problem/potential for technical effect

Who applie(s/d) which theory (when)

- Core theory: The "old" one
- Later: German supreme court discovers the "technical contribution" and assess inventions as a whole
- EPO "jumps" at this and continues to expand this theory
- German slowly reduces the patentability and moves slightly back towards the core theory
- EU SW patent directive:
  - → Moves through various iterations of various theories!
- Today most countries and the EPO follow the theory of holistic interpretation and require technical considerations
- EPO: Technical solution to technical problem
  - → Improved processing speed, economical memory usage, better UI etc.

The software patent directive: 2002-2005

- A EU directive to harmonize the patentability of SW
  - → Promoting innovation
  - → "Should not change the current state at all"
    - – I.e., codify what the EPO currently does
      - » Many would (and could/did) argue against this very seriously!
  - → Harmonization is an actual problem
    - » National differences; less in law than in decisions
  - → Unify European patent law with USA patent law
    - » But no not follow the USA in business method patents!
  - → European Court of Justice as supervisor
    - » Add a separate instance for "checking"
  - → Only patents on "technical" things
    - » But no definition what is "technical"!

Division of stakeholders

- Proponents: "Legal professionals"
  - » "We know what we talk about, we are the experts"
  - → Patent attorneys: More applications and cases
  - → Patent offices, EPO: Funding through applications and grants
  - → (Very) Large corporations: Protecting their developments
    - » And problems for new competitors
- Opposition: "Technicians/Programmers"
  - » "Software patents are the end of the world"
  - → Open source movement: Generally against IPRs/monopolies
    - » But rely enormously on copyright!
  - → Small companies: No cross-licensing → Pay or die!
  - → Individuals: "Crushed" by large patent holders

Stakeholders

http://www.boingboing.net/2005/07/08/prosoftwarepatentant.html

Timeline of the directive

- 2002: EU commission issues a draft
  - → Very liberal: "technical contribution"
- 2003: Passed by parliament, but with many modifications
  - → Very restrictive: "Technical = physical processes"
  - → Interoperability clause
- 2004/05: Parliament + Council of Ministers
  - → Compromise; parliament changes mostly lost
  - → Dutch parliament voted: Their minister has to change his vote
  - → Poland stated it could not agree to this position
  - → Legal affairs committee of EU parliament voted for a restart
- 2005: Legal affairs committee accepts council version
  - → Very narrowly only
- 6.7.2005: EU parliament reject the directive
  - → Without considering the 175 proposed amendments
  - → The first time ever of a rejection at the second reading

Argument for the directive

- SME only lack familiarity with patents
  - → When they know about it and there is a common legal framework they will profit from them too
- Incentive for research of new products
  - → Also for production and marketing → Only then some revenue
- Encourages disclosure
  - → Distributed the new ideas to the world instead of keeping them as company secrets (see EU vs. Microsoft!)
- Enables small companies to defend against large ones
  - → Patent infringement is very expensive (RIM vs. NTP)
- Asset to prove R&D to customers and financers
  - → Useful to gain venture capital of loans
- Required by TRIPS

Argument for the directive

- If you invent something, you deserve exclusive ownership
  - → You invested your work and energy → It should belong to you
- Protection for software patents is already sufficiently limited
  - → Trivial patents may have been issued, but lifetime is expiring
- Drugs would not exist without patents → SW is similar
  - → Both are difficult and expensive to develop, but easy to copy
- In SW most of the effort is in the programming, not the ideas
  - → Which idea would not have been created without patents?
- Copyright doesn't afford enough protection
- In the USA there are software patents → They are the best ones in software development → So they work
- Similar legal rules as in the USA
  - → Encourages EU companies to apply for US patents too

Arguments against the directive

- Expensive, uncertain and time-consuming to obtain a patent
  - → Copyright is "free"; patents require money else spent on R&D
- Prohibitively expensive in litigation
  - → And there need not always be a base for an accusation…
- Problem of independent inventions
  - → Independent writing → No copyright problems!
  - → No legal predictability
    - » Note also the 18 month till the publication of an application!
- Increases development costs: Searches and legal advice
  - → Extremely difficult to ascertain whether software infringes any patent, merely difficult and expensive to check a single one
- Duration of protection must be 20 years
- R&D is much simpler and cheaper in software
  - → "Everyone" is doing it → How many people research drugs?

Arguments against the directive

- Copyright is sufficient protection
- Software industry is working extremely well and has enormous growth, even without software patents
- Disclosure doesn't help: Patents are almost unintelligible for software engineers and programmers
- Standards might be covered by patents
  - → You have to pay to be able to support a standard
- Interoperability of systems may be reduced
- Open source is at a specific disadvantage
  - → Easier to find patent infringement
  - → More difficult to obtain patents
- Software doesn't need manufacturing: Copying is "free"
  - → What is fine for physical goods need not necessarily be so for immaterial ones

Reasons for failing

- "… Accordingly, inventions involving computer programs which implement business, mathematical or other methods and do not produce any technical effects beyond the normal physical interactions between a program and the computer, network or other programmable apparatus in which it is run shall not be patentable. "
  - → What does this exactly mean?
- Finally, everyone was afraid of the outcome that might occur through final voting on all the various amendments, and therefore voted against it
- Additionally, the resulting text was so vague that probably no harmonization would have resulted at all
- "We are now better off than when continuing this discussion"
- Also, some "irregularities" occurred during the procedure and some persons were "suspected"
  - → Family member involvement in campaigning for one side, decision must be yes otherwise this could be a precedent, …

Alternative approaches

- EPLA: European Patent Litigation Agreement
  - → Rendering patent prosecution more cheaply and easily
    - » Creating new patent courts which would replace the national patent courts for EU patents
      - – Allows extending the liberal EPO standard to all of Europe
  - → Perhaps already dead, as there are competence problems!
    - » Only the EU may institute it, not the member states
    - » Cannot be "inside" EU, as EPC includes other countries as well
- Community patent: A single patent for the EU (1960-???)
  - → A "real" EU patent, unlike the EPC
    - » Unlike EPC, no "national phase" any more!
    - » Would also include a single EU court for infringement/validity
  - → Very much under debate currently, but little progress
  - → Obviously requires some definition what is patentable…
  - → Main obstacle: Translations (count, time, validity, …)

Exemplary patents Overview

- Vicom: One of the first EPA decision on software patents
  - → "Mathematical method" vs. "manipulating image pixels"
- Anti-lock braking system: German decision "inventing" the core theory
  - → If a program is involved, a system may still be technical
  - → But it must employ controllable forces of nature
- SOHEI: Connecting two management systems
  - → A UI may be technical
- Computer program product: "further technical effect"
  - → Programs are patentable if they bring about a technical effect going beyond the "normal" physical interactions between the program (software) and the computer (hardware)

Vicom

- Improving a digital image through applying a matrix operation on each pixel and its surroundings
  - → No "forces of nature" to be seen anywhere!
  - → But "technical considerations" are obviously present
    - » The method would not work for audio signals at all!
- At first: Patent applied for the method → Denied
  - → Then: Method applied to images → Granted (prelim.)
- But: What about an analogue device doing the filtering which is controlled by a computer?
  - → The actual problem would be creating the "analogue device", not in the "controlled by a computer"!
    - » Matrix multiplication → Mathematical method
    - » Analogue device performing an equivalent → Patentable
  - → Removing "noise" from signals has always bee patentable
- Ultimately refused: Lack of novelty or inventive step

EP79300903 (A1); T 0208/84

Anti-lock braking system

- A method consisting of mechanical, electrical and electronical elements for regulating brakes
    - » This includes a computer program
  - → The rules for braking are not rules for thinking: They require the use of predictable and controllable forces of nature
    - » If you brake to hard, locking and skidding will occur
  - → Because of employing forces controlled through a computer in a specific way certain technical actions result
  - → Whether an invention is technical or not cannot be measured by its formulation; the content of the invention is decisive
- Theoretically, the ABS could also be constructed as a mechanical device → It would still brake identically and would undeniably be patentable
  - → The new and inventive part is how to brake, not doing this by computer (although without it might be impossible!)

BPatG 12.6.1978, AW (pat) 78/75

SOHEI

- If the solution requires some technical thoughts, then the invention has at least implicitly technical character
- Connecting two systems through using a single form on the screen to update two databases (inventory and billing)
  - → It implies handling files with different types of information
    - » Not technical are:
      - – The financial or inventory management
      - – The meaning of the data or the transaction details
    - » Technical features are:
      - – The unitary format of a "single transfer slip"
      - – The file management features made possible by the unitary format
      - – Through storing the data entered in a journal the processor always knows where exactly to find data to be copied to the databases. This allows updating various files directly from the stored transfer slip without involving the operator, obviating multiple inputs.

EP0209907 (B1); T 92/0769

Computer program product

- Only the claims 20 and 21 were under discussion
      - » I.e., the claims 1-19 were accepted already previously
  - → 20: Computer program product (CPP) loadable into memory performing the steps of claim 1 when run on a computer
  - → 21: CPP stored on a computer usable medium
- All computer programs modify the currents within the CPU
  - → This is the "normal" interaction of program and computer
- Technical can only be, what is "more" than this interaction
  - → Execution of the instructions can cause this
      - » Generated effect has technical character
      - » Software solves a technical problem
          - – Improved speed, less memory consumption, …
- No decision, but those claims are not generally excluded by "as such" → Examiner must check for such effect

EP0457112 (B1); T 1173/97

Computer program product

- "Definitions" from the decision:
  - → "Running on a computer": System comprising of program plus computer carries out the protected method
  - → "Loaded into computer": Computer is capable of carrying out the protected method
- Regarding "as such" it doesn't matter whether a program is claimed by itself or on a carrier
- Why are such claims interesting?
  - → Possession of a CD with the program is different from executing the program!
    - » Protected method is not executed when copying the medium
    - » Claim on medium prohibits this step/possession of such a CD

Austria: Software patents

- Input technical? → Works on image data from a satellite ✓
- Output technical? → Controlling robots ✓
- Technical means required (even when goal non-technical!)? ✓
  - → Text processing program finding spelling errors through a fuzzy-logic processor
- Non-technical aspects can never be part for "inventiveness"
- Mathematical methods are never technical
  - → A method can be protected for an application (VICOM), but remains free for use in other areas
- Information for the human intellect is not technical
  - → System for clustering taskbar buttons
- No claims on "programs" – only on "methods" & "procedures"
- Claims on "program on medium" are allowed

Software: Patents vs. copyright

- Copyright protects the independent creation; patents might still be infringed
- Copyright has a much longer duration
  - → Death of author + 70 years ⇔ ≤ 20 years
- Patents must be registered and require expenditure
- Patents are checked before granting, but almost all programs will qualify for copyright protection
- Patents cover not only the expression, but also the method implemented through the program
- Copyright protects "sweat", patents "genius"
- Patents must be disclosed, software can be distributed compiled and obfuscated

Software patents in the USA

- Everything can be patented as long as it is useful, concrete, and tangible
  - → This includes business methods, games, and software
  - → Mathematical methods are not patentable, unless combined with a specific practical usage
- Basis: Cases "Diamond vs. Diehr" (1981)
  - → About 1990 patentability of software was clearly established
  - → State Street Bank (1998): Business methods
    - »Everything except laws of nature, natural phenomena, and abstract ideas
- Many cases of successful prosecution of infringement
  - → Eolas: Browser plugin
  - → RIM vs. NTP (Backberry): Push E-Mail

Software patents in Japan

- Software patents, and business methods, are patentable
  - → But both require a "further technical effect beyond the normal interaction between soft- and hardware"
    - » Merely computerizing a mental or economical method is not sufficient for patent protection
- Unpatentable: mathematical methods or algorithms, learning methods, programming languages, information display, …
  - → Unless there is such a further technical effect
  - → Similar to Vicom: Interpolation method does not characterise the electrical characteristics of a real circuitry and does not employ the physical properties of such → Not patentable
    - » "Method for simulating a circuitry"

International comparison

- USA: Useful, concrete, and tangible
  - → Almost no limits at all
- EPO: Requires some technical effect
  - → Very broad; technical application sufficient
- Japan: Requires a "further technical effect beyond normal interaction between software and hardware"
  - → Similar to the proposed EU software patent directive
- Austria: Technical problem and technical means
  - → Very few decisions, so no definite answer possible
- Germany: Technical problem, solution, and means
  - → Alternatively: Technical problem
    - » Currently about equally divided: 50% decisions according to core theory, 50% according to holistic interpretation

Summary

- Main difficulty with software patents:
  - → When you should solve a problem, how probable is it to independently reach a solution which violates a patent?
    - » Actually a problem of triviality!
  - → Main idea of patents is to prevent "knock-offs" (economy) and ensure publication (society)
    - » Whether these aims can be reached by software patents is not very clear in my opinion
- No clear interpretation of laws or international consensus
- Could perhaps be only a transitory problem: Until all the trivial and "basic" software patents have expired

Software patents are not a legal discussion,
but really an economic, respectively political, decision!

Questions?

Your patent advisor is always there for you!

For individual discussions and other questions:
E-Mail: sonntag@fim.uni-linz.ac.at
Telefon: +43(732)2468-**9330**