COMMENT ON THE NOTE FROM THE EUROPEAN PARLIAMENT'S RAPPORTEUR ON COM(2002)92

ABSTRACT. This is a comment on a note from Arlene McCarthy, the Rapporteur of the Committee on Legal Affairs and the Internal Market (JURI) on the software patents directive COM(2002)92 (procedure 2002/0047). The note was meant for circulation in the conference in Brussels May 7th and 8th 2003 **[16]**, and has been circulated elsewhere. A PDF version of the rapporteur's document is at
http://swpat.ffii.org/events/2003/europarl/05/contrib/amccarthy/amccarthy030503.pdf

CONTENTS

The indented parts are quotations, the indented parts marked with sans serif like next paragraph are from the rapporteur document, the indented parts with serif font *like this parapgraph is from other sources as noted. Full width text are our comments.*

Note From the European Parliament's Rapporteur, Arlene McCarthy MEP

The Proposal for a Directive on the Patentability of Computer-Implemented Inventions.

1. WHAT IS THE CURRENT SITUATION?

The patenting of computer-implemented inventions is not a new phenomenon: patents involving the use of software have been applied for and granted since the earliest days of the European Patent Office (EPO).

It may well be that patents involving the use of software have been granted since the earliest days of the European Patent Office. They certainly weren't called "computer-implemented inventions", though. This term was introduced in 2000 by the EPO**[1]**. In EPO's youth it was clear that an invention could not be "implemented" in a computer. Let's be precise about terminology because that will allow us to identify a confusion that may be the source of much of the disagreement.

An invention is a new, inventive (non-obvious), industrial-applicable and statutory (not excluded) teaching. In a patent there is a description of the invention and a collection of claims. The claims define the monopoly that the patent grants, and must include the invention, but usually will include many other things which are not new, inventive, etc. but are needed to put the invention to work (for instance if you invent a new shape for a plane's wings you may want to claim planes with those wings, or the wings adapted to accommodate engines or landing gear, or the wings made of known materials, or whatever). You are not patenting all these known things you use to put your invention to work, it is only their use to realise your invention what is monopolised, so even if they appear in the claims, they are not patented as such (you can still use engines, wheels, materials in wings with other shapes, so it is not the wheel as such which is patented). What characterises the claimed object is the invention, not any accessory needed to make it useful.

In this sense, the claimed objects since the beginning of the EPO may well have included computers and software. But programs for computers is excluded subject matter (EPC 52), and therefore it can only be an accessory in the claims, it is explicitly disallowed as an invention, so that it can not be patented as such. You can, for instance, patent an industrial chemical process if you find some new and inventive way of mixing reactives under some controlled conditions. The control of this conditions may involve a computer and software, but you cannot claim the software as such. That is, it is not allowed to claim any unspecified industrial process that may be controlled by a computer with some particular new program. The program may be new, in a sense non-obvious, and be usable in industry, but it is not an invention, so it needs to be claimed, if at all, as an accessory of some invention.

This means that the program itself will not be patented, although it may be mentioned in the claims. So it is necessary to discern what is an invention from what is not, when examining the claims. This is clear enough in EPC 52, and was explicitly regulated in the EPO's guidelines, in their "earliest days" (1978)**[6]**:

> A computer program may take various forms, e.g. an algorithm, a flow-chart or a series of coded instructions which can be recorded on a tape or other machine-readable record-medium, and can be regarded as a particular case of either a mathematical method or a presentation or information. If the contribution to the known art resides solely in a computer program then the subject matter is not patentable in whatever manner it may be presented in the claims. For example, a claim to a computer characterised by having the particular program stored in its memory or to a process for operating a computer under control of the program would be as objectionable as a claim to the program per se or the program when recorded on magnetic tape.

> Applications are on the increase not only to the EPO but also to national patent offices in member states. Fifteen per cent of all applications for patents relate to computer-implemented inventions. Out of over 110,000 applications received at the EPO in 2001, 16,000 will have dealt with inventions in computer-implemented technologies.

Indeed, very impressive. Patent inflation is really a concern. Bessen & Hunt **[2]** show that strategic, anticompetitive and defensive use of patents tends to concentrate in software patents, because they are easier to obtain (they don't require experimentation or prototyping, not even writing a program). They are also broader, because software is not subject to physical constraints and can therefore be composed into more complex systems, potentially infringing in hundreds of patents per program. This causes a patent buildup similar to a cold war arms race that discourages innovation and competition, and instead of bringing new products to consumers, reduces their choice and their access to information society, resulting in significant costs and less productivity for businesses.

Given the current imbalance of costs and rewards in the EPO, where granting a patent brings revenues and rejecting it brings complaints and extra work in justifying the rejection, it is too easy for major corporations to build case law by simply drowning the EPO in applications for excluded subject matter.

So it is important to resist the pressure in order to keep a healthy knowledge economy. The best option would be, as the Economic and Social Committee of the EU suggests**[7]**, to reform or replace the EPO. A more immediate way, though, is to have the current EPC enforced to ensure that software is not patentable, and any software patent granted by the EPO is consistently invalidated. This may reduce the incentive to apply for excluded subject matter, and will of course bring legal certainty to developers and entrepreneurs.

> Concern has been raised that with certain EPO and national court rulings, if matters are left as they stand, Europe may be drifting towards extending the scope of patentability to inventions which traditionally would not have been patentable.

Moreover, concern has been raised that the EPO has already drifted a long way towards extending the scope of patentability to achievements which traditionally would not have been called inventions, and hence are not patentable**[3]**. Examples of improperly granted patents abound[4]. As Piia Noora Kauppi MEP said in Brussels 2003.5.7 "The EPO has been running wild". The expansion of subject matter is evident when comparing the 1978 EPO guidelines**[6]** quoted above with the current 2001 EPO guidelines**[5]**:

> [...]While "programs for computers" are included among the items listed in Art. 52(2), if the claimed subject-matter has a technical character, it is not excluded from patentability by the provisions of Art. 52(2) and (3). [...] But if a computer program is capable of bringing about, when running on a computer, a further technical effect going beyond these normal physical effects, it is not excluded from patentability, irrespective of whether it is claimed by itself or as a record on a carrier. This further technical effect may be known in the prior art.[...]

Which means that if you want to patent any software concept nowadays you just have to claim it has a "further technical effect" (which is defined to include either "technical considerations" on how a computer works, "technicality" in the meaning attributed to the data the software handles, or the efficiency and security of the process it describes).

Since any program for computers can be presented as having some efficiency or security trade-off, any software innovation can be patented.

If the program has any usefulness, so that its data can be said to refer to the real world, it will also be deemed to have "further technical effects". This policy misses the point of what does society get in exchange for the monopoly it grants, because it is very difficult to make sense of an EPO doctrine that amounts to saying that a program that adds 3 litres of fuel in a tank and 3 litres of fuel in another is technical, while a program that adds 3 EUR in an account, and 3 EUR in another account is not. The program is exactly the same, requires the same investment, and the only difference is what you apply it to.

If you don't want to deal with physical data or allege efficiency or security trade-offs to patent software you can still take a shortcut by wording your application with reference to elements any computer has (such as

memory, display, processor, storage means or other devices) and you are making "technical considerations", which gives your program for computer "technical character".

> [...]On the other hand a computer system suitably programmed for use in a particular field, even if that is, for example, the field of business and economy, has the character of a concrete apparatus, in the sense of a physical entity or product, and thus is an invention within the meaning of Art. 52(1) (see **T 931/95**, **OJ 10/2001, 441**)."

In other words, anything you can do by programming a computer is patentable, since you can claim a computer system programmed in some way. Since the computer system needs not be specified, and the characterisation may focus on the software, this means you can monopolise the use of any software by patenting a generic computer system with the software.

> [...]In the practice of examining computer-implemented inventions, however, it may be more appropriate for the examiner to proceed directly to the questions of novelty and inventive step, without considering beforehand the question of technical character. [...]The presence of such a technical contribution establishes that the claimed subject-matter has a technical character and therefore is indeed an invention within the meaning of Art. 52(1).

This clarifies that the EPO will interpret Art 52 in a manner equivalent to not having any exclusion for "programs for computers" at all. They have decided to apply the tests of novelty, inventive step and industrial applicability and disregard the test for whether there is an invention.

There is even a manual on how to patent any software concept**[8]**, and abundant granted software patents**[4]** confirm the lack of practical limits. Any program can be claimed to be more secure than some other, or to use less resources, and of course anything you write a program for can be claimed to be more efficiently done with computers than manually (otherwise you wouldn't write the program). But if we accept this criteria we are refusing any limits on subject matter and instead turning to usefulness, which is the US situation, maybe spiced with some additional rethorics about (non innovative) technical means.

The confusion arises because we desist in trying to assess what new teaching society gets from the patent and concentrate in the final result. When a programmer optimises a program so that it runs faster, or is more efficient, she is simply looking at what operations are not necessary to solve the problem, what results can be reused, or what organisation of data simplifies its use. She is not finding a way to etch smaller transistors in semiconductors so that the computer can perform more operations per second, or a better material to dissipate generated heat. There is no technical device running faster, there is simply less use of the same technical device.

There's a story about a famous mathematician that was punished in school when he was a child. The teacher had him add up all the numbers from 1 to 1000 so that he would not disturb for a while. Very shortly the child came up with the answer 500500, and the teacher was outraged, thinking he made that up, until the child explained it was very easy, look: 1+1000 = 2+999 = 3 + 998 = ... = 500 + 501 = 1001. So he didn't have to calculate 999 additions (1 + 2 + 3 +...), he could simply calculate one addition giving 1001, and then multiply by 500, because he saw there were 500 additions giving the same result. Did he use any brain accelerating drugs?. A faster pen for writing partial results? No. He "simply" found a way to do less operations, at the same speed as usual. If he had used a calculator instead of pen and paper, the EPO would probably deem the calculation speedup as a technical effect beyond normal interactions of arithmetic operations and a calculator, and would have granted a patent on it. But this is what programmers do daily in their jobs, so granting a patent for efficient software would be the information society equivalent of granting a patent for turning bolts in an engine assembly line in the industrial society. It is similar to claiming that driving a known car through a shortcut between two towns is a car boost and should be patentable, since it not only has the technical effect of moving people around, but also it has the further technical effect of reducing trip time.

This should be sufficient to show there is little fear that the EPO may drift towards granting patents for anything more you can do with computers, since there is little more left to include, but the current practice deserves concern. And this is not just a matter of legal tradition but of economic and social policy which should be based on sound understanding of what computers and software really are and how development

and research are done.

> If the EU does not take the step to develop its competence with regard to computer-implemented inventions, then the EPO and its boards of appeal will continue to be the main arbitrators of the law. This will side step the democratic scrutiny of the EU.

Very possibly so, indeed. The EU needs to take the step to democratically scrutinise the EPO**[10]**. This means the practice of the EPO must be questioned and whatever is legislated on software patentability must come from an informed inquiry and debate. Legislation should build on the decision by the democratic legislator of what is good or bad for the European economy and values. Accepting the current scope of patentability in the EPO practice, as the CEC proposal and the JURI rapporteur draft report do, without thorough economical, social and human rights justifications would be evading the responsibility of the democratic scrutiny, or even hinting that the EPO is welcome to undemocratically set the rules by drifting of its practice and subsequently expect the EU to endorse the fait accompli.

2. WHAT ARE THE AIMS OF THE PROPOSAL?

> To harmonise and clarify the law relating to patentability of computer-implemented inventions.

Harmonisation would be a noble goal if only the law was different in different member states. However the substantial patent law of the member states is already completely unified by the European Patent Convention, by which all the EU member states are bound and whose Articles 52-57 have been literally incorporated into the national laws. There may remain differences in related law (like fiscality of patents, etc.) but not in patentable subject matter. Any difference in subject matter is caused by the particular interpretation the EPO makes of the unified law, which conflicts with some courts interpretations**[9]** and is followed by others. These differences can be observed even within the same member states.

Yet clarity is important, since the more clear the law is the less room for differing interpretations (assuming rationality and good will in the interpreters). But one may wonder how much clarity the CEC proposal provides when the documents published by the Commission with the directive included a questions and answers document**[11]** which, when discussing the controversial "one click shopping" patent granted to Amazon.com, says that patenting this would be "highly unlikely" under the proposed rules (due to lack of inventive step), but that it does not want to rule absolutely on this matter because the EPO is studying it. This casts doubt on the success in any intention to legislatively control the EPO. Instead, it would suggest that the new criteria are so ambiguous that not even those who propose them are capable of applying the rules to an example they themselves have chosen.

Law experts say the directive does not clarify anything. See for instance the Bakels & Hugenholtz study **[12]** (or Bakels speech in Nov the 8th conference in the European Parliament, saying that the lack of clarity in the directive was bad for both pro software patent people and people against software patents, and good only for lawyers who could profit from the confusion).

The wish for clarity is thus very necessary indeed. And it can only be fulfilled by amendments which define terms that are central to the criteria in the directive, but remain undefined in the CEC proposal or the JURI rapporteur's draft report (such as amendment 45 defining "technical", amendment 51,52 and CULT amendment 14 explaining inventive step/technical contribution), or other amendments**[21]** that put forward clearer criteria that would set up more meaningful limits.

> To stop expansion of the patents system, and stem the current drift towards broadening the scope of innovation in software that can be patented.

Currently the only part left for this drift towards absolute patentability of software is providing the missing legal basis. As explained above, patents are already granted routinely by the EPO for all kinds of software innovations[4]. The directive proposal, by borrowing from the EPO practice, would open the floodgates of software patents enforcement.

The goal of stemming the drift from the EPC would be better served by amendments 32,33,41,43,45,46,48,51,52,59 and some others**[21]**, which reinstate the exclusion of programs for computers (and hence data processing) from patentability, and impose a true limit, uses of forces of nature in an inventive way. The only sound way to separate patentable from unpatentable subject matter is to look at whether the innovation is of deductive or empirical nature, i.e. whether it is logical or material, so that teachings that require experimentation, laboratories and prototypes are covered by patents and copyright covers the result of composing solutions by logical combination of operations defined in a formal model.

> To ensure that patents for computer-implemented inventions are granted on the same footing across the European Union and that national courts can deal with contested patents on the basis of uniform principles and within an EU legal framework.

As someone put it in the European Parliament conference, **2003.5.8**, "it is tradition when levelling a playing field, to remove the dirt, not to add more". Harmonisation is a worthwhile goal (which in this case can be more effectively attained by control on the EPO than legislation, but anyway). But harmonisation is no excuse to generalise bad policy or to cease to consider public interest when passing directives.

So in case the paragraph is meant to imply software patents pass from being of uncertain validity to being homogeneously enforceable, as the CEC proposal and the JURI's rapporteur draft report would bring, we should keep in mind that software patents have proved harmful to the economy and society**[2]**, are rejected by most of the software business and academics**[13]**, and are inconsistent**[14]**. So harmonisation is to be sought in the enforcement of clear, sound and economically justified limits to patentable subject matter.

The distinction between material and immaterial innovations outlined above would provide a consistent fixpoint where the practice of all patent offices could converge, instead of a wandering sequence of EPO case law eluding any systematic evolution, which the member state courts are asked to follow and the elected representatives to legalise after the fact.

> To create an EU directive which has the advantage of providing a high degree of legal certainty and uniformity across the Union. A body of authoritative, binding European case law will also be able to buildup, since the Court of Justice will have jurisdiction to give preliminary rulings allowing accountability and enhanced transparency of decision making.

This is a legitimate goal, but care is needed to achieve it. For the Court of Justice (which is already going to be competent by virtue of the Community Patent anyway) to be able to ensure a high degree of legal certainty and uniformity the legislator must give it a clear and unambiguous law. Merely endorsing unlimited patentability with a requirement of rethorical application crafting is of little use to any court. Those proposals will only generalise uncertainty in software business and, according to Bakels, a long delay until cases are resolved while courts struggle to make sense of undefined concepts. As long as the criteria allow software patents, the legal uncertainty of the software developers and users will be increased, since they'll have no way of knowing whether they are infringing on someone else's patents.

In the US, the creation of a specialised court to deal with patent cases (the Court of Appeal of the Federal Circuit, CAFC) is regarded as an important factor in the patent inflation that followed**[26]**. In order to prevent patent system problems as are currently felt there, since the EU is about to create one specialised court too, it would be wise to do so in a legal framework and a political attitude sensible to economical, social and human rights considerations.

3. How important is software to the EU economy?

> Software development is a major European industry: in 1998 the value of the EU software market was EUR 39 billion.
>
> Computer programmes are now used almost everywhere, for example for operating and controlling washing machines, lifts, automotive gear boxes and video recorders, as well as in the core areas of information technology, digital data processing, data recognition, representation and storage.

Indeed, almost anyone depends on computers today, and there are millions of professionals devoted to developing, maintaining and administering software. Most of these professionals (70% in Catalonia**[15]**) work in small or medium enterprises, which have shown deep concern on the CEC proposal**[16]**.

This is an important reason to take the utmost care in any legislation that may affect such an enormous market, such a widespread infrastructure and such an strategic EU policy (e-Europe, etc.). For a legislator to alter the information society landscape care should be taken to fully understand all the issues. Neither intuition nor easy analogies nor one sided interests should be used to rush uninformed decisions. Time should be spent on independent inquiry to understand the market and the nature of software and computers if there is honest interest in preserving and improving the EU advantage and opportunities in knowledge economy.

Inventions engendered as a result of R&D in the field of software are important to the EU economy.

Innovations engendered as a result of R&D in the field of software are important to the EU economy. Note that the pickiness in talking of innovations instead of inventions is not arbitrary. If one considers results of R&D in software to be inventions, then one is denying the technical character of inventions, that is the fact that they should provide new insights on uses of controlled forces of nature. Software does not provide these new insights, it merely assumes it will be run in some hardware that has already mastered control of forces of nature to present a reliable formal model to the programmer. This is as evident as saying that using a calculator may require knowledge of mathematics, but not of electronics, therefore a calculation with a calculator is not an invention, no matter how new, unobvious or industrial it is. It is important not to surrender one of the four tests a patentable invention must pass: technical character, industrial application, novelty and inventiveness (non-obviousness).

Removing the first test is opting for unbounded subject matter, just as in the US. Software may pass the other three tests but it will never pass a proper test for technicity. Having said that, yes, innovations engendered as a result of R&D in the field of software are very important to the EU economy. And understanding wealth production in the software economy is important to rightly regulate the field. If patents are useful to the industrial society is because they incentivate innovation, when the profitable activity is production, so that innovation might suffer without the incentive.

In the knowledge economy, or the software business, innovation is the rule. The only way to compete is by developing better software. Production and distribution of copies is equally cheap for all players. The only way to survive without innovation is having a monopoly, such as the one a software patent grants. So granting patents for software is the equivalent in the information society of what in the industrial society would be giving away monopolies in exchange for building a factory or even for tightening bolts in an assembly line.

These are already profitable activities in the industrial economy that do not require the privilege of a monopoly as incentive. Private property and labour laws are sufficient to ensure the factory or the work done can be sold in a just market. Just like the productive activity does not require special privileges, software innovation does not require software patents, only copyright, to cover the work done.

But this is only an analogy and intuition. The true proof of the harm caused by software patents is in the economical literature and the test case of the US patent system. Bessen and Hunt **[2]** show empirical evidence of a 10-15% decrease in R&D intensity in the US since they started to grant software patents. Bessen and Maskin provide a theoretical explanation of this behaviour**[17]**. The US Federal Trade Commission hearings**[18]**, news clips and other sources provide anecdotal evidence and quotations**[19]**. The European Software Patents Horror Gallery**[4]** proves that criteria as those in use by the EPO and proposed in the CEC text and the JURI rapporteur's draft have resulted in the same kind of broad, blocking and unnecessary patents in Europe than in the US.

The opposition of around 95 % of SMEs in the CEC Consultation[20] clearly indicates that the innovators in the European software economy are clearly asking "please, protect us from patent protection".

> At a time when many of our traditional industries are migrating to Asia, and when we Europeans are having to rely on inventiveness to earn our living, it is important for us to have the revenue secured by patents and the licensing out of ideas.

This statement is difficult to understand, and its relation with the directive is unclear. Patents do not discriminate by nationality of the applicant, so foreign companies can and do patent in the EPO. In software, most of the already granted software patents that the rapporteur's draft would legalise are in the hands of US and Japanese companies.

If the intended meaning is that international competitiveness will be improved by a more efficient European knowledge economy, then this is obvious, and another reason to keep software out of patentability, and use economic reasoning instead of dogmatic analogies to decide on legislation.

4. WHAT SORT OF PROGRAMMES WILL THE DIRECTIVE COVER?

> Thousands of patents have been granted by the European Patent Office and by Member States' national offices for technical inventions which involve computer programmes: currently as many as 15% of patents granted involve software.

This has already been commented above. But it's unclear how must it be interpreted under the title "What sort of programmes will the Directive cover?". There is no explanation on any criteria defining patentable programs, only a suggestion that lots of patents for software are already being granted, which might be assumed to mean the intention is to keep granting the more software patents the better. Must the lack of answer to the rapporteur's own question be interpreted as "almost any program will be patentable"?

The paragraph speaks of technical inventions. If the intended meaning is no software will be patentable as such, and only technical inventions (new teachings on using controlled forces of nature) are to be patentable, then this would be an endorsement of the amendments commented below **[21]** which reinforce and clarify the EPC. Note that either in the CEC text or the JURI rapporteur's draft or the paragraph above, the notion of "technical" is undefined, so insisting on it does only ask for approval of the amendments defining it, which is a welcome suggestion.

> During this time all sectors of the software industry, including the open source movement, have developed strongly. Patents are available only for programmes that offer a new technical solution.

We can agree to the prosperity of the European software business, including free software (or open source software if you like). In spite of disturbances caused by too simplistic spectations and unfounded valuation of immaterial assets in the .com crash, European software creativity is in good shape. Despite the documented anecdotal evidence of harm caused by European software patents granted**[22]**, the general understanding of their invalidity, backed by some courts, and the small number of court cases may have not been able to slow down very much the pace of progress.

But current prosperity in a sector is hardly a reason to change its current laws. The anecdotal European evidence and the US experience strongly suggested that a move to validate current and future European software patents would only aggravate the situation, and bring a new wave of unfounded speculation this time on inflated industrial property titles.

European software businesses would be better served by a clarification that software patents are invalid and should not be a cause of concern. This would lead to higher legal security and confidence in copyright over one's own work, higher investment (as testified by a venture capitalist in the European Parliament, Laura Creighton**[16]**) and a more competitive environment, unimpeded by any remaining intimidation power of currently granted software patents. This is what this sector is demanding.

5. WHAT SORT OF PROGRAMMES WILL BE EXCLUDED?

> Programmes which produce an aesthetic outcome, components of programmes(code sequences) which of themselves do not deliver any technical solution, algorithms and underlying programming ideas will not, as now, be patentable.

Again we see the term "technical" without a clear definition. Programs can never deliver a new technical solution by themselves, if "technical" is meant to have the delimiting meaning of use of forces of nature that it has and should have in patent law. Programs which produce an aesthetic outcome may not be patented because of lack of interest by the author, but there are a lot of what the EPO would call "technical considerations" involved in such a program (graphic compression, efficiency optimisation, or simply claims naming generic computer equipment). So they could not only be patented if desired**[23]**, but they are likely to infringe many already granted software patents. The CEC proposal or the JURI rapporteur's draft report would not change this.

A code sequence is not interesting to patent because it is too narrow. You can appropriate a code sequence by copyright, using patents for that would only be more expensive and have a shorter lifespan. Software patents claim a broad range of possible code sequences characterised by some new rule of calculation or organisation, not simply a code sequence. It is this broadness what makes them useful for anticompetitive strategies (besides the advantage of being much more cheap since they don't require empirical research).

Note that there is no way of limiting this broadness while keeping some usefulness for applicants. In really technical inventions broad claims are possible but they are restricted somewhat by physical constraints. In software there is no limit to composing solutions so that any innovations can be used in a great many uses, often quite unrelated to the original intention, but possibly covered by clever claim drafting. This broad applicability of any idea is what makes broad software patents inevitable. Software is the art of abstraction. Software patents will therefore be either very broad or very trivial, often both.

The possibility to patent inventions and keep ideas not covered is an exclusive property of the material world. In technical invention patents what is monopolised is a physical object or process, and what is published is a description of that invention. In software patents, the claim is characterised by programs, that is, by information. A program is its own description, it's only information. So it belongs to the publication side, not to the monopolisation side. Attempting to make information both a disclosure and a claim raises contradictions. Programs, algorithms and ideas have the same nature. They are all information. The only difference is in the amount of detail in the expression of the information. A complete idea is an algorithm (which is equivalent to a mathematical theorem), and an algorithm fully specified for a computing environment is a program. The more you try to avoid patents on ideas and algorithms, and accept patents on programs, the more detailed the claim would be, until you end up with the scope copyright gives for programs and patents become useless. So exclusion of algorithms and ideas is infeasible for software patents.

We'll try to exemplify it with an analogy to a model of programming that might be familiar to more people: spreadsheets. Assume you have the idea nobody has had before of using a spreadsheet to keep an account of the stock in your warehouse. Your idea is keeping information on the number of items of each article, and its mean price of purchase so that you know its value. When you buy more stock you should enter the amount of each item bought and the price you bought it at. Since you know the total price of your current stock, you can add the price of the new stock and divide the total price by the new number of items in stock to have a new mean price per item. You have all the data, so it should be possible.

That's an idea. An algorithm would be very similar. You should decide whether you put articles in a row or a column (let's say a column), what data you keep (let's decide you have one column for the name of the article, another one for the number of items, another for its mean price and another for its total price). In an algorithm you should specify that the total price is calculated by multiplying the number of items by the mean price per item. Then you can add two more columns for items bought and their price. You would finally specify another column is going to receive the value of (stock total price + items bought * price bought) / (stock items + stock bought).And then there should be a way to substitute the current stock by the current stock plus the items bought and the current price by the new result.

The program would be written by opening your spreadsheet, deciding what column goes where, typing in the formulas in the particular spreadsheet language, copying the formulas to the particular range of rows, and maybe adding a button to update the current stock with the calculation. You should add the names of articles and the corresponding numbers, one by one, and test it. This spreadsheet, and small variations on it would be covered by copyright. You could remove the test data and sell licenses for your spreadsheet. People could not copy your spreadsheet without your permission.

But other people may solve the problem in a different way. For instance, they could decide to add a column for the new stock items, or one for the total price of the newly bought stock. They could do it without the column for the current stock total price. There are many ways to do it, and each would possibly lead to an independent creation that is not covered by your copyright.

So you could go to the EPO and obtain a patent for a "method and apparatus for automated stock value computation" or something like that. In the patent claims you would possibly want to claim "a computer system having storage means to store amount of items in stock and its price for a plurality of items, coupled to input means to enter purchase data to be displayed in a terminal screen, and calculation means to update said items and said data stock so that the new mean price of purchase is displayed". A few more claims would prevent people from either copying your spreadsheet or writing their own, since whatever they did would infringe on some claim. So you would be monopolising your idea of using a spreadsheet (or in fact a computer) to calculate stock prices. In fact there is a similar French patent already granted, so whatever novelty and inventiveness it may had, it seems clear that this is currently accepted as patentable subject matter in European patent offices. It is not important to discuss whether this is "protecting" the idea, the algorithm or just a set of formulas or mathematical operation sequences. The fact is that nobody can put the idea to use, so it is effectively monopolising an idea and an algorithm. If it wasn't it would be a waste of time and money for the patent holder.

This would block any future stock owner from using her computer and spreadsheet to perform that computation, no matter whether they have ever heard you had the idea before, or whether they have copied anything. You might even collect some royalties from it (unless the developer of the spreadsheet had enough patents to block you when you start being a problem for their business). But society would not have gained any innovation that would not have been there without the patent. Faced with the problem of knowing the stock value and having a computer around, anybody would have sooner or later implemented the idea with the same costs as you. In fact anybody would do well to buy you a license for your copyrighted spreadsheet, if you sold licenses for less than the development costs (which would be profitable for you, since you'd sell many of them).

No big investment is needed to come up with the idea because once you read the computer or spreadsheet manual and see that there are columns, rows and cells, and you can add, multiply and divide the numbers in those cells, you know you can solve your problem. This is how software works, the manual (the formal model of the computer) tells you the operations available and that makes ideas easy (passing from ideas to programs is harder, that is what makes copyright valuable). Nothing to do with discovering a new chemical or a medicine, where laboratories, experiments and essays are necessary before you have an interesting idea application to patent, because we don't have the manual of the real world. And in that case the distinction between the chemical substance or the process of production and the description of it, would make it easier to ensure that pure ideas are not patented, but only concrete applications of them.

In summary, if you don't want ideas or algorithms to be patentable, you don't want software patents, you want copyright. It would be interesting to look at the current EPO guidelines**[5]**, some software patents granted by the EPO**[25]**, and the CEC proposal or the JURI rapporteur's draft report to try to find what in the new directive could be used to stop ideas or algorithms from being patented, given that the concept of technical invention (new teaching on use of forces of nature) is abandoned.

> Programmes which underpin non-technical inventions will also not be patentable, in particular those concerned with non-technical business methods (such as"reverse auction" arrangements) along with the methods themselves. Although these are widely patented in the United States, in Europe it has been shown that there to be no case for such extension of patents in Europe.

"technical" is still undefined, so "non-technical" is undefined too. According to the patents granted by the EPO, "non-technical business methods" must mean something like a business method that excludes any use of computers. Direct applications of business concepts to computer and networked environments are granted patents by the EPO**[24]** following the same criteria in the CEC proposal or the JURI rapporteur's draft.

For instance one of the headlines about business method patents in the USA was

"Divine Inc. extracts dotcom tithe from UK firm"

http://www.theregister.co.uk/content/6/28803.html

Divine Inc. attacked Thompson and Morgan (an Ipswich company who made business in the USA) with patent **US5715314**. The EPO database gives an equivalent European Patent, **EP0803105**, on electronic commerce with a bank payment gateway. Thompson and Morgan has not been sued in the UK, as far as we know, but the chances for Divine Inc. to try it in the UK too will increase if the technicity criteria in the EPC is dismantled as per the CEC proposal or the JURI rapporteur's draft report. It may be interesting to ask whether we welcome such news or what provisions in the directive proposal can use a judge to invalidate this patent

http://l2.espacenet.com/espacenet/bnsviewer?CY=ep&LG=en&DB=EPD&PN=EP0803105&ID=WO+++9613013A1+I+

Some more example of (technical, according to the JURI rapporteur) business method patents:

EP792493: B1 Dynamic prices
http://swpat.ffii.org/pikta/mupli/ep792493/index.ca.html
EP762304: B1 Electronic brokering
http://swpat.ffii.org/pikta/mupli/ep762304/index.ca.html
EP807891: B1 Electronic shopping cart
http://swpat.ffii.org/pikta/mupli/ep807891/index.ca.html
EP756731: B1 Generating buying incentives from cooking recipes
http://swpat.ffii.org/pikta/mupli/ep756731/index.en.html

These patents have been granted under EPO criteria contrary to the EPC like the CEC proposal and the JURI rapporteur's draft report propose. Declaring these innovations "technical business methods" will not help the economy nor the companies that will have to pay royalties for no good reason.

> The EU directive should therefore provide a restrictive statement of the law. Software programmes that produce merely an aesthetic outcome, a component of programmes (code sequences) which themselves do not offer a technical solution, algorithms, underlying programming ideas and programmes that underpin non-technical inventions, should not be patentable.

The EU directive can still provide a restrictive statement of the law, for example if the right amendments**[21]** are voted and others are not. Stating desirable policy goals but failing to ensure them in the draft report is a mistake that can still be corrected.

6. What would the situation be without the EU directive?

> As current law is not clear, uncertainty and lack of transparency would remain. This uncertainty is a brake on growth in the industry, and is allowing the scope of what is patentable to expand, contrary to wishes expressed throughout the EU.

Without the EU directive the situation would be the current one, which was reasonably presented as in good shape above. Since the written law is clear and the EPO's case law is not binding on national case law, EPO-granted patents may still continue to be revoked by national courts, the EPO itself will be under pressure to reform.

The directive as proposed by the CEC would consolidate the EPO trespassing into unpatentable subject matter, and encourage further expansion beyond computers by fait accompli politics. The JURI committee rapporteur's draft report would not make the situation any better.

The directive is difficult to amend since it is based in fundamentally wrong assumptions, and includes many redundant provisions to erode the delimiting criteria mandated by the EPC. Yet the three European Parliament committees involved have made a good job together, and there are tabled amendments**[21]** that could solve the problems and improve clarity and juridical certainty for European Citizens, by reinforcing the EPC and helping EU judges invalidate software patents.

So in case the right amendments**[21]** are all voted and the wrong ones are all rejected, the directive could indeed improve the situation to some extent. Concessions or deviations from this baseline, with very few exceptions, could lead to stagnation of the currently healthy European software sector. So fixing the directive is indeed the best and necessary option, but the only alternative to a real, sound and clear fix is rejection.

> Most patents in Europe are granted by the European Patent Office (EPO) who, facing well-drafted applications and the need to offer applicants the benefit of the doubt, are contributing to a slow but discernible drift towards wider patenting.
>
> Failure to adopt this directive will allow this to continue and will deny the patent authorities in Europe a clearer platform on which to operate. Failure to adopt will also mean that the EPO may act by itself, thus side-stepping the debate and democratic scrutiny of the EU route.

Devaluating the debate and democratic scrutiny of the EU by dogma, weak arguments and threats of administrative malpractice, or ignoring the economic evidences, stakeholders view, and basic values would be even worse than keeping the EU uninvolved. The legitimity of the EU cannot be put in doubt by uninformed decision making, disregard for public opinion or interest or blind following of any institution. At the very least, any proposed legislation should be tried with some example patents and the benefits of the outcome should be justified**[25]**.

> With regards to calls for abolishing, within the EU, all patents on computer-implemented inventions, EU companies would be at a severe disadvantage in the global market place if they were not able to apply for a patent over their invention.

Maybe the confusion in this paragraph is caused by encompassing inventions where software plays an accessory role in teachings on new uses of controllable forces of nature and patents where the only innovation is in software under the term "computer-implemented inventions". The EPC and some tabled amendments**[21]** separate these two well enough. Since there is no big pressure to abolish patents on new uses of controlled forces of nature than happen to be claimed with some accessory software, we must assume the paragraph refers to calls to abolish software patents, patents where there is no invention and the only innovation lies in rules of organisation and calculation like software.

This is almost the current situation, where software patents are granted but seldom enforced, and the law does not allow them, so it seems a little alarmist that reinforcing the current situation would lead to sudden competitive disadvantage for the EU. If the competitive disadvantage was true, then the goal of restrictively restating the law or keeping any limits in patentability would have to be discarded in favour of some "more patents, more competitive advantage" theory.

But anyway, the argument is misleading, since it assumes a priori either some positive effect of software patents or some asimetry in access to foreign patents. Since patents are only valid for the territory where they are granted, in case Europe stays clear of software patents and the USA keeps granting them, European software developers and companies will have a safe home where they'll be able to grow and work without being attacked with software patents, and at their option they'll be able to file for USA software patents and use them against their competitors there. European companies will only be vulnerable to attacks with software patents if they have businesses in the USA. On the other hand USA companies will be vulnerable at home. This would be a competitive advantage for the EU, although both markets would use the same rules for all players, either national or foreign.

7. WHAT INTERNATIONAL AGREEMENTS AND CONVENTIONS NEED TO BE TAKEN INTO ACCOUNT?

> The EU has obligations under international agreements and conventions that deal with patenting of computer-implemented inventions. Two such obligations one must keep in mind are the European Patent Convention (EPC) and the Trade-Related Aspects of Intellectual Property Rights (TRIPS).
>
> Under Article 52(1)-(3) of the EPC, and hence under national law, all patentable inventions must be novel, involve an inventive step and be capable of industrial application.

And programs for computers are not inventions at all. Not being an invention means that software cannot be patented as such, although the presence of software in something else which is an invention does not render it unpatentable.

> Additionally, Article 27 of the TRIPs agreements states;
>
> "Patentable Subject Matter "
>
> 1. Subject to the provisions of paragraphs 2 and 3, patents shall be available for any inventions, whether products or processes, in all fields of technology, provided that they are new, involve an inventive > > step and are capable of industrial application. Subject to paragraph 4 of Article 65, paragraph 8 of Article 70 and paragraph 3 of this Article, patents shall be available and patent rights enjoyable without discrimination as to the place of invention, the field of technology and whether products are imported or locally produced.
>
> 2. Members may exclude from patentability inventions, the prevention within their territory of the commercial exploitation of which is necessary to protect order public or morality, including to protect human, animal or plant life or health or to avoid serious prejudice to the environment, provided that such exclusion is not made merely because the exploitation is prohibited by their law.
>
> 3. Members may also exclude from patentability:
>
> (a) diagnostic, therapeutic and surgical methods for the treatment of humans or animals;
>
> (b) plants and animals other than micro-organisms, and essentially biological processes for the production of plants or animals other than non-biological and microbiological processes. However, Members shall provide for the protection of plant varieties either by patents or by an effective sui generis system or by any combination thereof.
>
> The provisions of this subparagraph shall be reviewed four years after the date of entry into force of the WTO Agreement."

In case this quotation is meant to suggest that software must, by virtue of the TRIPs agreements, be considered in a field of technology as the CEC proposal would, Bakels and Hugenholtz come to mind. They are not the only experts saying so, but respect for public money justifies a quote from the study**[12]** commissioned by the JURI committee itself (page 14 and 15):

> Proponents of software patenting have argued that Article 27(1) does not allow software from being excluded from patentability, since computer software is to be considered a "field of technology". The discussions preceding adoption of the TRIPs agreement, however, do not confirm such a reading. In the absence of a legal definition of "invention", the agreement arguably leaves it to the member states to determine what constitutes a patentable invention, and whether or not that includes computer software as such.
>
> [...]
>
> The TRIPs agreement requires patent protection to be available in "all fields of technology". Arguably, this does not mandate patent protection of computer programs per se. Surely, TRIPs does not prescribe patentability of business methods.
>
> The generally perceived need for world wide patent law harmonisation should not lead to the conclusion that the European patent system must follow U.S. developments automatically.

8. How do patent and copyright protection differ?

> Copyright and Patents are two types of rights that compliment each another whilst having different functions and scope.

They compliment one another as long as they are applied to different objects. For instance a patent could apply to some peripheral hardware and copyright to its driver program. As long as you try to apply copyright to hardware or patents to software, you run into contradictions. If we had copyright and patents for software, like in the USA, we would see the same kind of weakening of intellectual property by erosion of copyright. Writing a program should entitle the author to control who and how can copy her program, sell licenses or otherwise profit from her job. Allowing third parties to patent concepts that the program writer may have employed means that it is no longer the program writer, but the patent owner who controls distribution, copying and use of the writer's program. Copyright is automatically enjoyed by authors, while patents are cumbersome, expensive and bureaucratic, by the way.

> A patent protects an invention and copyright protects the particular expression of a computer programme, ie the actual lines of code written by a programmer.

Copyright covers more than the actual lines of code written by a programmer. It covers any derived works, including modifications to the program, inclusion of parts of a program in another, linking a module of a program to another, etc.. It therefore covers the set of programs that would be generated by copying all or part of the program and maybe modifying it or building on it. But it does not cover so much variation that unwilling infringement is likely. Programmers know very well how to avoid liability by copyright infringement. They don't use other people's code without ensuring they have a license to do so. And they make sure the license for what they write is easy to find where the code is, so that others can respect their copyright. They can reasonably easily stay within the law or face the consequences if they don't.

Patents cover inventions, and inventions should require experimentation. prototype testing, etc. This makes more unlikely that the investment to arrive to a new invention is made without checking whether the solution has already been found, at least if the patent offices properly assess the four tests of subject matter, novelty, inventivity and industrial applicability. Also material constraints often lead to scenarios closer to one product one patent, so that keeping current with the patents in a field is a bit less of a burden.

More importantly though, patents are the only way to appropriate inventions. Inventions are generally hard/expensive to develop and easy/cheap to copy once the problem has been solved. The expensive part is experimenting to see how the forces of nature can be controlled.

Software is even easier to copy literally, that's why copyright is useful, but a developer using only someone else's ideas (not someone else's code) will probably need a similar effort to write the code than the originator of the ideas, so independent reimplementation is not a concern. On the other hand the ideas themselves (unlike implementation) are cheap, since software development happens inside a formal model of computation perfectly known from the start.

So software patents would "protect" the cheap investment (idea) and erode the property on the expensive investment (implementation, code). Copyright covers individual creation of the human intellect, such as programs, and patents cover empirical and applied R&D of the industry. They were not meant to overlap, and the increasing importance of software in our economy does only make more important that they don't conflict with one another.

> Copyright gives the right to prohibit the copying or commercialisation of that code as an intellectual creation, but does not protect the ideas underlying software, what the software does within a machine or how a machine, under software control, interacts with its environment.

From an earlier section it appeared that the intention was not to "protect" the ideas or algorithms, so what's the need for patents?. Ideas and thoughts are free.

"What the software does within a machine" is a bit unclear: software does nothing inside a machine, just like a recipe book cooks nothing inside a kitchen. Software is information on how to solve a problem with a computer, and the computer is a machine that reacts to some appropriate supports of that information according to certain rules that allow the writer of the information to ensure that the computer will solve the problem. But software only defines a process, never performs it.

The clause "how a machine, under software control, interacts with its environment" is more understandable. Yet since software defines that control, it defines the interaction, so copyright on the software would be enough to prohibit copying the software to the machine and therefore using it to control such interaction with the environment. If the problem is that this prohibition is not as broad as some would want we may be returning to the debate of whether ideas should be owned.

9. WHAT ARE THE OBJECTIVES OF THE REPORT?

> There is a need to ensure that patents for computer-implemented inventions are granted on the same footing across the European Union and that national courts can deal with contested patents on the basis of uniform principles and within an EU legal framework.

As already said, harmonisation or a legitimate grab of competence (e.g. bringing patent system under control of EU parliament) is no excuse to generalise bad policy or to cease to consider public interest when passing directives.

> An EU directive should not allow the extension of patentability, but neither should it exclude patent protection altogether. Small software developers should not have to face a minefield of poorly granted patents for obscure or obvious patents.

This desirable goal may be attained by passing the amendments**[21]** outlined above. Patentability would not be extended beyond what EPC already specifies, patents would still be available for technical inventions (new teachings on use of controlled forces of nature), regardless of whether software is used or not in the invention, and software developers and users should not worry for patents as long as the only thing they develop, publish or use is software.

> The directive must allow open source software to flourish. Indeed many large corporations rely on the inventiveness of small business. They too must be able to turn their creativity into returns on their investments. Small companies and open source software developers should be able to exploit their inventiveness. The report will include a review clause to ensure that the European Commission monitors and reports on the impact of the EU directive for SMEs and software developers.

Again desirable goals to be attained only by clear exclusion of software innovations from patentability, not by the CEC proposal or the JURI rapporteur's draft report. The monitoring would be better carried out by either an observatory as proposed by Bakels and Hugenholtz in the study commissioned by the JURI committee**[12]** or by a Parliamentary Committee, than by the same institution that proposed the directive. Once the independence of the monitoring is beyond all doubt, it will only be useful with an adequate budget and powers of inquiry, and monitoring should probably start before the directive is enacted, so that changes caused by the directive if any can be compared with the initial situation.

Today, none of the available studies, reports and opinions (including those from the JURI committee itself) support the CEC proposal or the JURI rapporteur's draft report. Consequently the same good intention to monitor and report on proposed legislation would be well used to take available reports into account in writing the legislation.

> With Europe having no powers to prevent a gradual drift towards the patentability of business methods as witnessed in the US, software developers only recourse then would be to bring proceedings

> in national courts which would only leave in place the present fragmented and uncertain environment for business and industry.

Certainly, it is too bad that European democratic institutions cannot more closely control such a perturbation of the European market as drifting patentability standards can provoke. To ensure that at least the recourse to the courts exists, the EU should make sure that the directive reinforces EPC by clarifying the exclusion of programs for computers as per the suggested amendments**[21]**.

The European Parliament can start this tighter control over the EPO with legislation reinforcing a clear limit for patentability, and later see how the EPO could be reformed or replaced so that the current imbalance of incentives biased towards patent inflation can be corrected.

REFERENCES

[1] Trilateral website document (European, USA and Japan patent offices) document introducing the term "computer-implemented invention"
http://www.european-patent-office.org/tws/appendix6.pdf

[2] An Empirical Look at Software Patents James Bessen (Research on Innovation and MIT), Robert M. Hunt (Federal Reserve Bank of Philadelphia) May 2003
http://www.researchoninnovation.org/swpat.pdf
Other economic bibliography:
http://www.softwarepatenter.dk/economics_articles.html
http://swpat.ffii.org/vreji/prina/index.en.html
http://swpat.ffii.org/vreji/minra/sisku.en.html
http://swpat.ffii.org/vreji/papri/index.en.html

[3] Erosion of the concept of technical invention
http://swpat.ffii.org/analysis/invention/index.en.html

[4] European software patents horror gallery
http://swpat.ffii.org/vreji/pikta/index.en.html

[5] EPO examination guidelines, 2001Chapter IV, point 2, title "programs for computers"
http://www.european-patent-office.org/legal/gui_lines/e/c_iv_2.htm

[6] On the EPO evaluation guidelines from 1978
http://swpat.ffii.org/papers/epo-gl78/index.en.html

[7] Opinion of the Economic and Social Committee of the EU on the software patents directive **CES1031-2002**
http://europa.eu.int/eur-lex/pri/en/oj/dat/2003/c_061/c_06120030314en01540163.pdf

[8] Patenting Software Under the European Patent Convention Keith Beresford Sweet & Maxwell; **ISBN: 0752006339**
http://www.smlawpub.co.uk/products/cat/mydetails.cfm?title=5414&detail=5414

[9] For example, see the decision of German courts in 2002
http://swpat.ffii.org/papers/bpatg17-suche02/ or
http://www.aful.org/wws/arc/patents/2003-05/msg00094.html

[10] As amendment 2 in either ITRE or CULT opinions point out

[11] Questions and answers document by CEC
http://www.europa.eu.int/comm/internal_market/en/indprop/02-32.htm

[12] Bakels and Hugenholtz study for the JURI committee
http://www.europarl.eu.int/meetdocs/committees/juri/20020619/SoftwarePatent.pub.pdf
Comments on it http://patents.caliu.info/dgiv.html

[13] Opposition to the directive
http://patents.caliu.info/explicacio.en.html#L118

[14] Leaflet on software patents
http://wiki.ael.be/index.php/TheDangerOfSoftwarePatentsToEurope

[15] Statistics on Information and Communications Technology sector from the Generalitat de Catalunya (Catalan government) say that for those companies with more than 10 employees, around 70% of jobs are in companies with less than 200 employees. http://www.gencat.es/csi/pdf/cat/estadistiques/Ofer

[16] As was evident in the Brussels events of May 7th and 8th 2003
http://www.greens-efa.org/fr/agenda/detail.php?id=998&lg=fr
http://swpat.ffii.org/events/2003/europarl/04/index.en.html

[17] James Bessen & Eric Maskin Sequential Innovation, Patents and Imitation, Working Paper, Department of economics MIT, Cambridge, Massachusetts. English
http://www.researchoninnovation.org/patent.pdf

[18] Federal Trade Commission hearings on patents and competition
http://swpat.ffii.org/papers/ftc02/
http://www.ftc.gov/opp/intellect/index.htm

[19] Interesting quotations on software patents
http://swpat.ffii.org/vreji/quotes/index.en.html

[20] On the CEC consultation on software patents
http://swpat.ffii.org/vreji/papri/eukonsult00/

[21] List of necessary amendments 4, 5, 9, 17, 18, 20, 22, 23, 24, 25, 26, 27,28, 31, 32, 33, 34, 35, 39, 40, 41, 43, 45, 46, 48,50?, 51, 52, 54, , 59, 63, 65, 68, 70, 71. Cult4, Cult5, Cult6, Cult9, Cult11, Cult18, Itre1, Itre2, Itre3, Itre11, Itre14, Itre15, Itre18, Itre19. For more details on why these are needed and what other options are there, and what are the consequences of each choice, see
http://swpat.ffii.org/papers/eubsa-swpat0202/juri0304/

[22] Effects of software patents
http://swpat.ffii.org/patents/effects/index.en.html

[23] Some patents already come quite close to aesthetic programs:
EP689133 Graphics elements
EP663089 Virtual reality information presentation

[24] Quotes provided by Carsten Svaneborg

The First-Mover Monopoly, Likhovski, Spence, og Molineaux. http://www.oiprc.ox.ac.uk/EJWP0500.pdf

> Except: The law in relation to business method patents in the United Kingdom and European Patent Offices has not changed. Away of doing business as such cannot be protected; however, some protection for business methods may be obtained by claiming a new, inventive, technical method of implementing a business method. This is particularly relevant in the eCommerce arena where there have been and continue to be technical innovations.

K. Beresford, World Patent Information 23 (2001) 253-263 European patents for software, E-commerce and business model inventions.

> Abstract: It is commonly held that software, e-commerce and business models related inventions are inherently unpatentable in Europe. In reviewing the situation, this article emphasises the large number of inventions in this field which have in fact been patented through the EPO or through national patent offices in Europe. [..] He shows that, if proper attention is paid to both the substance and the form of claims and description, to direct the reader to the technical problem and its solution, effective protection is in fact very often available. He gives the following examples:

Examples: business management and financial systems:

EP407026 (Reuters)
EP209807 (Sohei)
EP701717 (Shepherd)
EP838063 (Realkredit Danmark AS)

(The last one was subsequently retracted. Because the Realkredit DK patent was a business method, they were sued by a competitor claiming it to be invalid, but the case and the patent were both dropped when the competitor and Realkredit later merged.) Some more examples:
http://www.iusmentis.com/patents/businessmethods/epoexamples/

[25] Patentability Legislation Benchmarking Testsuite
http://swpat.ffii.org/analysis/testsuite/

[26] Article by Brian Kahin comparing the USA and EU situation in software patents
http://www.firstmonday.dk/issues/issue8_3/kahin/index.html