position on the proposal for a directive on the patentability of computer implemented inventions
Although most of the parties involved in drafting the current proposal for a Directive on the patentability of computer implemented inventions stated that they want to avoid the patentability of software, we are convinced, that the current proposal does not effectively safeguard against it.

1. Why patents on software have to be avoided:

1. There is no need for further protection

The basic theory of the patent system is simple and reasonable. It is desireable in the public interest that industrial techniques, which are not protected by copyright, should be improved. In order to encourage improvement, and to encourage the disclosure of improvements in preference to their use in secret, any person devising an improvement in a manufactured article, or in machinery or methods for making it, may upon disclosure of the improvement at the patent office demand to be given a monopoly in the use of his invention for a period of 20 years. Hence, a patent should serve as an incentive for improving the known art and sharing the improvement with the public domain, by giving a monopoly for using the patented item.

Software, however, is explicitly protected by copyright, so nobody may lawfully use a copy of software without a licence of the author of such software. A similar protection is also established by legislation against unfair competition.

Additionally, the investments involved in software development processes are far below the costs of development processes in the traditional industries, accordingly the protection of such investments with a patent is disproportional.

2. There is no need for further incentives for improvement

No one could possibly say, that the development in the software/IT-industry is too slow or that ideas are not quickly disseminated. On the contrary, the software/IT-industry is among the fastest developing branches of industry.

3. Patents on software would inhibit improvement

However, many do say, that the factual domination by the big players, which already is in place, works against improvement. Only the big players in the software market have the necessary funds for carrying out a large number
of patent proceedings. Therefore the factual domination of the big players would be enhanced if they also get the opportunity to establish legal monopolies, which will last for 20 years. This will inhibit improvement severely.

Lycos is very active in software development, as it requires less resources than in traditional industries. However, this will change when software becomes patentable in the EU. As patent law does not allow for parallel inventions, we would have to carry out extensive patent searches in order to avoid patent infringement. Out of Lycos’ own experience, in the US, where software effectively is patentable, a patent search can easily cost 30,000 USD. And in the event possibly infringed patents are found, one has to assign patent-experts with scrutinizing whether one’s own development would infringe such patents. After that, one can still not be sure to act lawfully and one could still be faced with long-lasting patent infringement proceedings. Taking this into account we would in many cases decide not to do our own development.

4. Patents on software would damage the software/IT-Industry

Patents are only granted for coherent inventions and anyone who prepares a patent application will try to formulate the patent claims as abstract from the intended application as possible, in order to have the widest protection possible. In the extreme, this will lead to the protection of algorithms for basic data processing tasks, if software becomes patentable.

Algorithms work for software applications like bricks work for houses. If one would give a patent on a certain kind of brick, one would effectively give a monopoly on building houses with such a brick, even if one wanted to build a house which is very different from the house the brick was invented for. Accordingly, if one would grant a patent for an algorithm one would give a monopoly on every application which makes use of that algorithm. This would have devastating effects, which may be exemplified by the following look at two algorithms which could be used on an average working day at Lycos Europe:

■ **Evaluation systems:** A popular example in this area is Google’s page-ranking algorithm. It is based on the extremely simple idea that, if a lot of websites refer to a certain website, then that website should be classified as particularly important. This concept is applied in many other fields and not only in connection with searches. The process is incorporated in an algorithm. If this algorithm became patentable, it would not only prevent further development of this idea in the area of search software, but would also, assuming that the patent specification would be very abstract, hamper development in other areas.
Spam recognition software: Spam is a significant problem in the area of email. All providers rely on being at liberty to actively confront this issue. The patenting of filters, which range from simple to very complex processes, could promote the further dissemination of spam.

For further examples see the Appendix: Specific consequences of Patents on Software – A look at an average working day at LYCOS

II. How to avoid the patentability of software

All parties involved in the drafting of the directive on the patentability of computer implemented inventions claim, that they want to avoid the patentability of software. Therefore the goal of preventing software patents is unanimous.

This was also the case, when the current European Patent Convention (EPC) was passed in 1973. Especially Article 52 sec. 2 c in conjunction with sec. 3 EPC states that programs for computers as such shall not be regarded as inventions and are therefore excluded from patentability.

However, it is estimated that around 30,000 pure software patents were already granted by the European Patent Office. Even if it is difficult to confirm this number, there are a few outstanding cases where trivial software patents were granted under the current legislation, e.g.:

- Patent on a Method of displaying multiple sets of information in the same area of a computer screen (EP0689133) – grants a patent on the method of putting windows on a computer screen behind each other and making them accessible with tabs like in a conventional paper index. The patent-owner successfully claimed 2.8 Billion USD from a party which used these tabs.

- Patent on a method in a computer system for ordering a gift for delivery from a gift giver to a recipient (EP0927945).

These are good examples of clearly non-technical and common solutions (e.g. tabs in a paper index) becoming “technical” and thereby patentable by simply applying them in a software solution. Accordingly, the exclusion of patentability of software in the EPC showed ineffective in excluding the most trivial software patents.
This shows how legal provisions are susceptible to interpretation. The current proposal for the directive is ambiguous in many aspects. These ambiguities open backdoors for the patentability of software and are likely to cause legal uncertainties, which will have to be resolved in costly court proceedings with unpredictable outcomes.

Lycos therefore deems it to be paramount that the European Parliament makes sure that any uncertainty within the directive as to whether software is patentable or not is avoided, by introducing unambiguous wordings which clearly exclude the patentability of software. In the following we will show the most ambiguous regulations within the current proposal with regard to the patentability of software, the effects of such ambiguity and which of the amendments submitted for the second reading, we deem to be most effective in preventing ambiguity.

### Name of the Directive and throughout the Directive

<table>
<thead>
<tr>
<th>Current proposal</th>
<th>Supported amendment</th>
</tr>
</thead>
<tbody>
<tr>
<td>patentability of computer-implemented inventions.</td>
<td>ability of computer-aided inventions.</td>
</tr>
</tbody>
</table>

### Justification

Software by its very nature is always computer-implemented. The term “computer implemented invention” therefore suggests that pure software could be a patentable invention. What is meant however, is that industrial engineering processes, which also make use of programs running on computers should be patentable. Such processes could only be “aided” but not “implemented” by or on a computer.

Accordingly, the term “computer-aided inventions” should be used in the title and throughout the text.
Article 2 – Definitions

Current proposal

(a) “computer-implemented invention” means any invention the performance of which involves the use of a computer, computer network or other programmable apparatus, the invention having one or more features which are realised wholly or partly by means of a computer program or computer programs.

Supported amendment by Pia-Noora Kauppi

(a) “computer-implemented invention” means an invention within the meaning of the European Patent Convention, the performance of which involves the use of a computer, computer network or programmable apparatus.

Justification

Pursuant to the proposal, one of the examples for a computer-implemented invention would read:

“... any invention the performance of which involves the use of a computer, the invention having one feature which is realised wholly by means of a computer program”

This suggests that a program running on a computer can constitute a patentable invention. By leaving out the term “computer program” this would be prevented.
### Article 2 - Definitions

<table>
<thead>
<tr>
<th>Current proposal</th>
<th>Supported amendment</th>
</tr>
</thead>
<tbody>
<tr>
<td>(b) “technical contribution” means a contribution to the state of the art in a field of technology which is new and not obvious to a person skilled in the art. The technical contribution shall be assessed by consideration of the state of the art and the scope of the patent claim, which must comprise technical features, irrespective of whether or not these are accompanied by non-technical features.”</td>
<td>(c) “field of technology” means an industrial application domain requiring the use of controllable forces of nature to achieve predictable results. “Technical” means “belonging to a field of technology”.</td>
</tr>
</tbody>
</table>

### Justification

The definition of “technical contribution” has to be clear in order to determine what is patentable and what is not. Reference to “technical features” is too vague as software in itself could be regarded to have technical features. The understanding of the word “technical” lies at the heart of the whole software patent issue. Accordingly, what is needed is a further clarification of the term “technical”. 
The inventive step should be in the technical contribution for the computer-implemented invention to be patentable.

One could read the second sentence of the proposal as a definition of the term “inventive step”. The wording of the proposal suggests that a computer-implemented invention simply by making a technical contribution may involve an inventive step. In order for a new computer program, executed on a technical device not to be a patentable invention, the inventive step has to be evaluated solely in relation to the technical contribution of the patent claim. Accordingly, it must be made clear, that the inventive step must lie in the technical contribution.
**Article 4 - Exclusions from patentability**

<table>
<thead>
<tr>
<th>Current proposal</th>
<th>Supported amendment</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) A computer program as such cannot constitute a patentable invention.</td>
<td>(1) Programs for computers are not inventions in the sense of patent law.</td>
</tr>
</tbody>
</table>

**Justification**

The exclusion of “software as such” is not sufficient, as it is unclear what differentiates software from “software as such”. The exclusion of “software as such” from patentability could lead to the conclusion that only software which has no apparent interaction with anything outside of itself and the computer it is running on is not patentable, which again leads to the conclusion, that software which does interact with anything outside of itself and the computer it is running on could be patentable. This is the very argument the EPO used when interpreting the exclusion of “programs for computers as such” in Article 52 sec. 2 and 3 c EPC, leading to the current legal uncertainty and the examples of patents being granted for trivial software, shown above. Why the new directive should make use of the same ineffective wording is incomprehensible. Instead one should make use of an exclusion which restates Article 52 sec. 2 EPC without the problematic “as such”-exclusion of Article 52 sec. 3 c EPC.
## Justification

The requirement that an invention involving a computer program must produce technical effects "beyond normal physical interactions between a program and the computer" in order to be patentable, is an invitation to interpret the whole directive in favour of software-patents, as the central term "normal" is utterly indefinable and can be interpreted in any direction. The term "normal" will invariably become a matter of debate in numerous legal proceedings where patents were granted based on this term. Therefore Article 4 sec. 2 of the proposal should be deleted.

The supported amendment offers a definition for "computer program" which is one of the central terms within the directive.
The aim of every programmer when writing software, is (or should be) to optimise the use of computing resources. The German Supreme Patent court found: “if an improvement of efficiency in the use of computing resources, such as time or data space, is deemed to be a technical contribution, then all computer-implemented business methods become patentable.” This shows that the term “beyond normal physical interactions between a program and the computer” in Article 4 sec. 2 – besides of being indefinable – is too narrow to effectively limit patentability of software, as it would not be effective when a new program (as a side-effect) makes more efficient use of computer resources such as energy and storage, because these are effects, which do not result from an interaction between program and computer. The amendment makes sure that a common side-effect of software development does not constitute a technical contribution.

Article 4 – Exclusions from patentability

<table>
<thead>
<tr>
<th>Current proposal</th>
<th>Supported amendment</th>
</tr>
</thead>
<tbody>
<tr>
<td>by Barbara Kudrycka, Tadeusz Zwiefka, Fausto Bertinotti Piia-Noora Kauppi, Manuel Medina Ortega</td>
<td>(3) Member States shall ensure that data processing solutions are not considered to be patentable inventions merely because they improve efficiency in the use of resources within data processing systems.</td>
</tr>
</tbody>
</table>

Justification

The aim of every programmer when writing software, is (or should be) to optimise the use of computing resources. The German Supreme Patent court found: “if an improvement of efficiency in the use of computing resources, such as time or data space, is deemed to be a technical contribution, then all computer-implemented business methods become patentable.” This shows that the term “beyond normal physical interactions between a program and the computer” in Article 4 sec. 2 – besides of being indefinable – is too narrow to effectively limit patentability of software, as it would not be effective when a new program (as a side-effect) makes more efficient use of computer resources such as energy and storage, because these are effects, which do not result from an interaction between program and computer. The amendment makes sure that a common side-effect of software development does not constitute a technical contribution.
Article 5 - Form of claims

<table>
<thead>
<tr>
<th>Current proposal</th>
<th>Supported amendment</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) Member States shall ensure that a computer-implemented invention may be claimed as a product, that is as a programmed computer, a programmed computer network or other programmed apparatus, or as a process carried out by such a computer, computer network or apparatus through the execution of software.</td>
<td>(1) Member States shall ensure that a computer-implemented invention may be claimed only as a product, that is as programmed apparatus, or as a process carried out by such an apparatus.</td>
</tr>
</tbody>
</table>

Justification

The word “only” has to be inserted in order to give any limiting effect to the article.

As in most cases computer programs are implemented on computers, the use of generic or common computer equipment should not be taken into account in favour for the patentability of a given invention and therefore should be avoided in the wording.
Justification

Pursuant to this article of the proposal a claim for a computer program, in connection with a claim for a process put into force by such computer program, can constitute a patentable invention. The term “process”, however, is very broad and not limited by any definition. All computer programs could be said to be putting into force a process (as in “data processing”). Therefore one could patent a computer program, simply by adding a claim to the process it puts into force. This results in the patentability of software. This is not wanted by any party, it is incoherent with regard to the other articles in the directive and it is not in the scope of “Article 5 – Form of Claims”.

Article 5 - Form of claims

<table>
<thead>
<tr>
<th>Current proposal</th>
<th>Supported amendment</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2) A claim to a computer program, either on its own or on a carrier, shall not be allowed unless that program would, when loaded and executed in a programmable computer, programmable computer network or other programmable apparatus, put into force a product or process claimed in the same patent application in accordance with paragraph 1.</td>
<td>(2) A patent claim to a computer program, either on its own or on a carrier, shall not be allowed.</td>
</tr>
</tbody>
</table>

by Barbara Kudrycka, Tadeusz Zwiefka, Fausto Bertinotti Piia Noora Kauppi, Evelin Lichtenberger, Monica Frassoni Manuel Medina Ortega
In addition to the amendments supported in order to prevent ambiguities with regard to the patentability of software, Lycos supports this amendment because we know, that interoperability of data processing systems lies at the foundation of the information economy and allows for fair competition by all players large and small.

Article 6 of the current proposal only refers to the exemptions provided for by the Copyright directive. This means that a software developer is allowed to find out how to make his data processing system interoperable with that of a competitor. However it remains unclear whether the working of an interoperable software which invariably makes use of the patented invention, would constitute a patent infringement. The amendment makes sure that a patent cannot be used to prevent the making and using of interoperable software.
Recitals

The recitals obviously need to correspond to the suggested amendments of the articles, as recitals are used for the interpretation of any article.

III. Conclusion

Lycos is convinced that the current proposal for a directive on the patentability of computer implemented inventions leaves room for software patents. Software patents would have a devastating effect on the whole software/IT-industry. As all parties unanimously want to prevent the patentability of software, all must agree in finding the wording which is most effective in order to reach this goal. We believe that the above suggested amendments are an important step in this direction and therefore, we strongly support them.

Appendix

Specific consequences of Patents on Software

A look at an average working day at Lycos

I. Examples at the product level:

Searching & shopping:

Search algorithms: Search algorithms allow large volumes of data to be searched quickly. In principle, the search function is based on basic computer science techniques, which every computer scientist learns as part of his studies. The algorithms employed constitute the “basic tool” used by every student to learn how the search function operates. Consequently, students use these algorithms during their studies, but they also apply them in practice. In addition, new and modified methods are constantly being developed as standards and requirements change, and they are quickly incorporated in computer science courses. If these algorithms were patented in future, future software developers would no longer be able to freely learn these processes in the course of their
studies (or learn processes that accord with latest developments) and could no longer resort to them because of the cost associated with their use. This is of fundamental importance for our “search” product, but also for each “content management system” and every other database whose use is made fast and efficient through the use of search algorithms.

**Classification functions:** One of the main functions of search and online shopping software is the classification function, which sorts and pre-classifies the vast range of data for the user. Like search functions, most classification algorithms are based on standard computer science techniques, which are enhanced depending on the relevant application. It would be extremely difficult to assess just what the existing standard algorithms would be and what could potentially be patented, and therefore it would not only be tedious to accomplish, but also not without an element of doubt. The patenting of individual parts of the classification function would therefore severely restrict the ability to use the “free” components as well.

**Evaluation systems:** A popular example in this area is Google’s page-ranking algorithm. It is based on the extremely simple idea that, if a lot of websites refer to a certain [site], then that [site] should be classified as particularly important. This concept is applied in the most varied of fields and not only in connection with searches. The process is incorporated in an algorithm. If this algorithm became patentable, it would not only prevent further development of this idea in the area of search software, but would also, assuming that the patent specification would be very abstract, hamper development in other areas.

**Encryption algorithms:** Various methods are used to encrypt emails to ensure that data can be securely transmitted via the Internet. These algorithms are often very complex. Nevertheless, email providers need the use of these algorithms to be cheap, because e.g. the increased use of additional computers solely for the purpose of encryption and decryption can have a substantial impact on the email provider’s cost structure. If encryption algorithms were patentable, competition among email providers would be severely restricted.

**Spam recognition software:** Spam is a significant problem in the area of email. All providers rely on being at liberty to actively confront this issue. The patenting of filters, which range from simple to very complex processes, could promote the further dissemination of spam.
Dating sites

Matching algorithms: Generally speaking, complex matching algorithms are used in all currently existing dating services. Most are based on the attributes of the user. Any patents in this area could rapidly become trivial patents, because all methods are actually based on trivial heuristic methods. Matching takes place on the basis of simple attributes, such as age and gender. If a matching algorithm were generalized, as patent specifications are wont to do, all matching rules would be affected, even if they were based on other attributes, such as special characteristics or hobbies.

II. An example at infrastructure level: billing

User scoring functions: In order to avoid fraud in connection with online payments, various user classification functions are used. These are based on general data and attributes, meaning that most algorithms implement simple heuristic methods. As in the case of dating sites, patenting in the area of user scoring without regard for the consequences could have a substantial impact on Lycos Europe's business, even though the algorithms involved are simple and extremely intuitive. The algorithms used in order to minimize the risk of fraud evaluate a user on the basis of his payment history or the validity of the credit cards used etc. Just like the matching algorithms used for dating sites, if a generalized scoring procedure were patented (and all patent specifications seek to generalize), all functions would be affected, even though they could be deduced intuitively.

III. Examples at the operating level: operating systems, networks & databases

Open source operating systems: Lycos Europe uses hundreds of computers in order to run its services. In this respect, we are reliant on (among other things) solid operating system components. Gaps in security must be closed quickly. Nevertheless, further development is necessary. We are able to provide some of our products such as free email, free web hosting and dating free of charge only because we have a reasonably priced but nevertheless reliable basis. This is ensured by open source software components such as Linux (operating system), Apache (web server), Java and PHP (programming languages) and MySQL (database). If this software were patentable, the development of these open source components would be jeopardized, and so would all of Lycos Europe's activities in all open source areas.
Security: In the field of security, Lycos relies on free and open dealings with all other enterprises whose work involves the Internet. Patenting in the field of security without regard for the consequences could have serious consequences in terms of the spread of viruses and Trojan horses etc. These may in turn paralyze whole segments of the Internet or individual companies for hours or days.

These examples particularly show that the algorithms that would ultimately be protected by patent form the basic tools used by software developers. Algorithms often implement a general concept. If this tool is no longer freely available for use and modification, software development as we know it will no longer be possible. This will have a substantial impact on software development and on the availability of free Internet services.

Status: June 2005