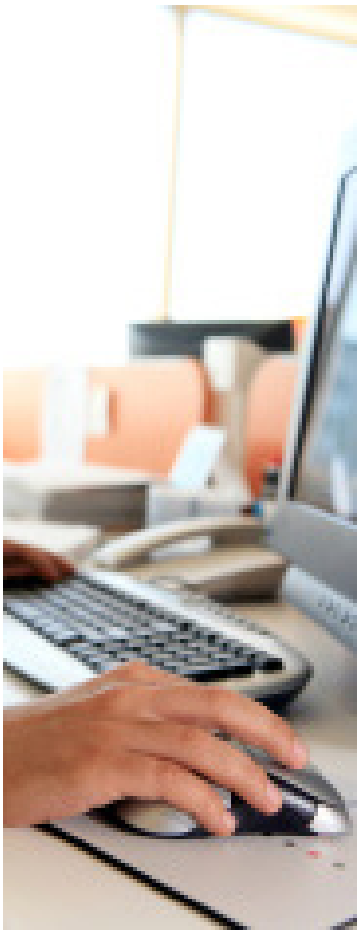
BakerHostetler

Brief

Using Structural Claim Limitations to Protect Software Patents



A major area of concern for clients with software patents is the way in which the claims are written, i.e., using arguably ambiguous and overly broad functional claim limitations, which can lead to uncertainty associated with the meaning and scope of the claims.

Some commentators have argued that patents do less good and cause more harm in the software industry than in other industries. Claims for patents often are written using functional, rather than structural, terms. It is frequently argued that such claims are overbroad and unclear, and that the aggregation of such claims can result in innovation-stifling patent thickets.[1]

1 Mark A. Lemley, Software Patents and the Return of Functional Claiming, 2013 Wis. L. Rev. 905, 906.

With this contentious issue in mind, our team advises clients on how structural claim limitations may instead be used effectively to avoid any potential issues created by the use of functional limitations in software patents. The proper use of structural claim limitations can save many a claim from invalidity on § 101 or § 112 grounds. The software field is somewhat unique, e.g., compared with chemistry and biotechnology, because there is no standard language for software patents. Accordingly, it can be difficult to know what a software patent covers until a judicial body, such as a court or the Patent Trial and Appeal Board, has construed the claims and issued a ruling.

The attack on software patents has been very effective as we've seen a dramatic increase in the number of patents invalidated based on the U.S. Supreme Court's decision in *Alice v. CLS Bank*, which significantly tightened the standard for what is and is not patentable. Since this decision was handed down in 2014, courts applying *Alice* are invalidating software patents at a previously unheard of rate under the so-called abstract idea exception of Section 101.[2]

Why functional claim limitations are used in software patents

There are two likely explanations for why functional claiming is so common in software patents:

1. As a matter of technology, a principal feature of computer software is that structure and function can be separated. A software developer can often write new software without knowing details of the hardware on which the software will run.
2. As a matter of language, a distinguishing characteristic of software is a lack of a commonly accepted "vocabulary" for defining software elements.

In software, a broad claim requires defining the invention at a higher level of abstraction. It is common for software developers to coin new terms to define the functional elements of the software, and the meaning and scope of such new terms is often not explicitly defined in the patent. This can make it very difficult for a reader to understand the precise scope of protection of the claims.

2 See Curiouser and Curiouser: Is 'Alice' the Long-Sought Troll Killer (The Legal Intelligencer) http://www.flatfeeipblog.com/files/2015/06/Curiouser-and-Curiouser-Is-Alice-the-Long-Sought-Troll-Killer-_-The-Legal-Intelligencer.pdf; and Guest Post: In Rush to Invalidate Patents at Pleadings Stage, Are Courts Coloring Outside the Lines? (Patently-O). http://patentlyo.com/patent/2015/07/invalidate-pleadings-coloring.html?utm_target/=feedburner&utm_medium=feed&utm_campaign=Feed%3A+PatentlyO+%28Dennis+Crouch%27s+Patently-O%29

Functional Claim Limitations and 35 USC § 112(f)

The law seems to be evolving such that purely functional claim limitations will be construed as § 112(f) elements, i.e., limited to the structure described in the specification and equivalents thereof. In addition, such claims will be limited by the scope of the written description, or invalidated as being indefinite under § 112(b) if insufficient structure is described. Accused infringers can now use § 112(b) and § 112(f) arguments to invalidate many patent claims that include a broad and largely functional recitation — a situation frequently encountered in claims involving software.

Software claims are typically drafted using functional terms; i.e., the claim recites what the software does rather than what it is. The software claim elements generally do not specify particular coding approaches or modules that must be used, much less the code that implements those modules.

However, structural limitations, whether in the form of algorithms or other constraints, can be utilized to strengthen software patents against attacks under § 112 as well as § 101. The logic behind this thesis is straightforward: First, the recitation of structure in a claim element should be sufficient to avoid means-plus-function treatment under § 112(f). Second, the recitation of structure in a claim necessarily constrains the scope of protection and arguably makes the claim nonabstract.

Structural Claim Limitations for Software

One could argue that a functional claim element need not be unclear/indefinite under § 112(b), or overbroad under § 112(a), and that an inventor of a software product that performs a novel/unobvious function should be able to broadly protect the product. Unfortunately for inventors in the software field, the courts and the USPTO seem to be taking the law in a different direction.

Computer software refers to organized collections of computer data and instructions, which can be divided into two major categories: system software and application software. The application software interfaces with the user and system software, and the system software interfaces with the hardware, as illustrated in Fig. 1.

Figure 1

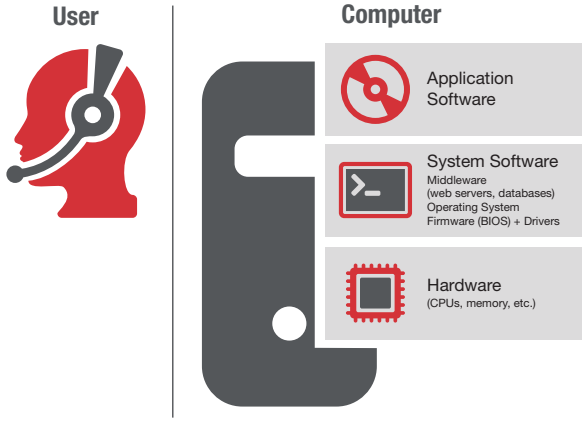
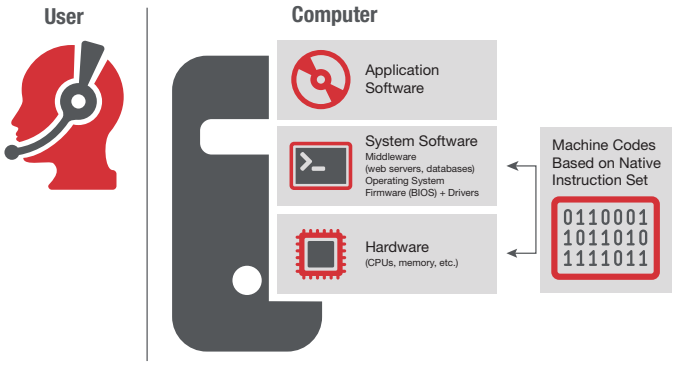


Fig. 1 is obviously an extreme simplification of a working computer system. An essential feature, which is not shown, is the necessary relationship between the executable software and the hardware processor.

The interrelationship between the executable software instructions and the hardware processor is structural in the sense that it greatly constrains any particular system implementation. In other words, the instructions per se are simply a series of symbols or numeric values. They do not intrinsically convey any information. It is the processor, which by design was preconfigured to interpret the symbols/numeric values, that imparts meaning to the instructions.

Figure 2



The foregoing concepts are illustrated in Fig. 2 and it is these concepts that can be used to prepare a "structural description" of computer software elements.

Any description of a software element in purely functional terms, i.e., in terms of what it ultimately does or the result it ultimately achieves, is unconstrained and therefore overly broad. The interrelationship between the executable software instructions and the hardware processor constrains the way in which the hardware processor is controlled in order to achieve the desired function/result. Problems related to abstractness, indefiniteness and overbreadth can be addressed by describing inventive computer software elements in the specification at this level of detail, and including appropriate limitations in the claims.

To be clear, we are not advocating inclusion of specific machine code or even source code listings in patent applications. (This would be unhelpful to patent examiners and the public, with the possible exception of copyists.) Instead, the technical/structural nature of the invention can be more precisely set out by describing the overall system architecture in the manner explained above. In addition, appropriate flow charts describing the logical operation of the software relevant to the inventive elements and technical features should be included.

Using the Structural Description of Computer Software for Claim Drafting

Now let's consider how claim 8 of U.S. Patent No. 6,155,840, at issue in *Williamson v. Citrix Online LLC et al* (Fed. Cir. 2015), might be redrafted using these ideas. Below is a copy of claim 8 with markings indicating how it might be rewritten:[3]

8. (Amended) A system for conducting distributed learning among a plurality of computer systems coupled to a network, the system comprising:

- a presenter computer system
- an audience member computer system
- a distributed learning server, comprising:
  - » a hardware processor configured to perform a predefined set of basic operations in response to receiving a corresponding basic instruction selected from a predefined native instruction set of codes;
  - » memory;
  - » a streaming data module; and
  - » a distributed learning control module comprising a first set of machine codes selected from the native instruction set for receiving communications transmitted between the presenter and the audience member computer systems, and a second set of machine codes selected from the native instruction set for relaying the communications to an intended receiving computer system, and a third set of machine codes selected from the native instruction set for coordinating the operation of the streaming data module, wherein each of the first, second, and third sets of machine code is stored in the memory.

The above version of the "distributed learning control module" element should not invoke § 112(f) because the first, second and third sets of machine codes constitute sufficient "structure" for the respective function, i.e., so as to avoid falling within the ambit of § 112(f). This approach to claim drafting should also be effective in avoiding *Alice* abstractness issues and overbreadth issues under § 112(a).

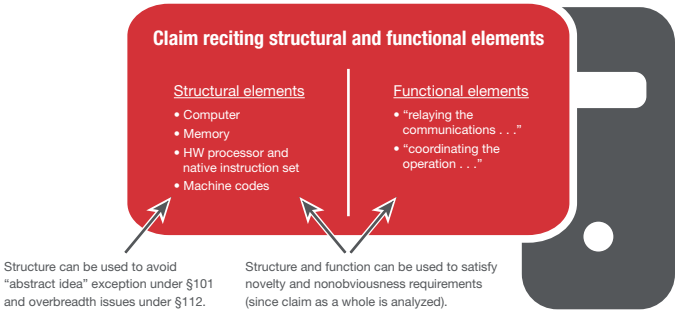
3 For this exercise, I assume that the specification provides support for these amendments.

A potential problem with the above solution is that an anti-patent cynic could argue that the limitations added to the claim are generic limitations inherent in any modern computer, and that such limitations should be ignored for purposes of §§ 101 and 112. However, the added limitations do make a difference, because they add structure sufficient to make the claim nonabstract under § 101 and also not purely functional, thus avoiding § 112(f). In other words, whereas functional claim limitations are now generally ineffective in avoiding problems under § 101 and § 112(f), structural limitations are, or should be, effective to avoid these same problems.

Moreover, both structural and functional recitations should be considered for patentability under §§ 102 and 103. This is somewhat analogous to the situation in Europe, where the dichotomy is not between structural and functional features, but rather between technical and non-technical features.[4]

4 In Europe, excluded subject matter is defined by Article 52(2) EPC. If a claim recites non-technical features together with at least one technical feature, then the EPO will not reject it for relating to excluded subject matter under Article 52(2) EPC. However, only technical features are considered for the novelty and inventive step. See Article 52 (2) EPC https://www.epo.org/law-practice/legal-texts/html/epc/2013/e/ar52.html; T208/84 (Vicom) https://www.epo.org/law-practice/case-law-appeals/recent/t840208ep1.html; and T931/95 (Pension Benefits) https://www.epo.org/law-practice/case-law-appeals/recent/t950931ep1.html.

Figure 3



Conclusion

Describing software elements in terms of "structure" may be a good alternative to an algorithm or functional description, or in addition to these. If included in a patent claim, a structural description of the kind described above may be sufficient to issue under §§ 101, 112(a) and 112(f).

Contact

Michael D. Stein
T 206.332.1384
mstein@bakerlaw.com