Missouri Law Review

Volume 76
Issue 3 *Summer 2011*

Article 8

Summer 2011

Patent Law's Unpredictability Doctrine and the Software Arts

Greg R. Vetter

Patent Law's Unpredictability Doctrine and the Software Arts

*Greg R. Vetter**

TABLE OF CONTENTS

* Associate Professor of Law, University of Houston Law Center; Co-Director, Institute for Intellectual Property and Information Law; biography available at: www.law.uh.edu/faculty/gvetter. My background includes a Master's degree in Computer Science and nine years full-time work experience in the software industry. My thanks to Seth Cockrum and Craig Walter for excellent research assistance. For helpful comments and discussion, I thank Chris Holman, Dennis Crouch, Jeanne Fromer, Kathy Strandburg, Mark Lemley, Lisa Dolak, Lee Petherbridge, and participants at these events: The Center for Intellectual Property Law and Information Technology (CIPLIT®) 9th Annual CIPLIT Symposium; *Cyberlaw 2.0: Legal Challenges of an Evolving Internet*, at the Depaul University College of Law; The *2010 Intellectual Property Scholars Roundtable*, by the Intellectual Property Law Center at the DRAKE UNIVERSITY LAW SCHOOL; The *2011 Works-in-Progress Intellectual Property (WIPIP) Colloquium*, at BOSTON UNIVERSITY SCHOOL OF LAW; and The Missouri Law Review 2011 Symposium: *Evolving the Court of Appeals for the Federal Circuit and its Patent Law Jurisprudence*, at the UNIVERSITY OF MISSOURI SCHOOL OF LAW.

TABLE OF FIGURES

I. INTRODUCTION

For software patents, much of the influence on the enablement doctrine and its undue experimentation proviso, including the important "unpredictable technology" factor for the proviso, arises from cases where the software technology at issue is now many decades old. For example, in the important software enablement case of *Northern Telecom, Inc. v. Datapoint Corp.*,[1] the original filing date of the claims for a data entry terminal was in 1971.[2] The terminal used software to operate, thus raising the question whether the patent gave sufficient information to allow an artisan to make and use the disclosed technology without undue experimentation, given that the source code was not disclosed.[3] The software was tied closely to the hardware, partly because its purpose was to give the terminal its functionality, and partly because software ran closer to the hardware in the early 1970s than it does today.[4] The more the software of the 1970s ran close to the hardware, the more it might have made sense to develop a jurisprudence categorizing software as a predictable technology, akin to many areas of electrical engineering. But software and the law relating to software patents have changed dramatically since the 1970s.[5]

Using a particular model of research and development, in light of the insights above, this Article argues that the unpredictable technology doctrine should not be applied categorically. The insights of the model suggest this conclusion generally, but the article treats software as an example of the issue.

The model postulates phases for a research and development project and then presents curves representing the learning effort typically observed under

1. 908 F.2d 931 (Fed. Cir. 1990).

2. U.S. Patent No. 3,760,375 (filed July 1, 1971).

3. *N. Telecom*, 908 F.2d at 941-43.

4. *See* Christine E. Reinhard, *Tangible or Intangible – Is That the Question? Conflict in the Texas Tax Classification System of Computer Software*, 29 ST. MARY'S L.J. 871, 875-76 (1998).

5. Thomas P. Burke, *Software Patent Protection: Debugging the Current System*, 69 NOTRE DAME L. REV. 1115, 1128 (1994).

each phase.[6] Developed by Peter Norden, an IBM researcher, the model remains influential in software development as a theoretical basis for software project cost estimating.[7] This Article refers to the model as the Norden model, using a stylized presentation of the model from a software engineering textbook.[8] The phases are planning, design, prototype, product, and modification.[9] The last phase, modification, is not used in this Article's application of the model.

In the Norden model, enablement is a relationship between the design and prototype effort curves. The research and development team will transition from the design phase to the prototype phase. Information transference will be comparatively efficient, assuming continuity for the research team. How difficult it is to build the prototype will depend in part on the quality of the design, and in part on the then-present character of the technology niche at which the research is aimed. Further, the current state of knowledge about the technology influences the success of design and prototyping. For successful research and development, the team works through the phases in a progressive fashion.

To map this pattern to enablement, the design information corresponds to the disclosure given in a patent instrument. The prototype is the effort of an artisan to make and use the invention. But a patent introduces an information discontinuity. Not only is the patent-reading artisan not an original developer, the nature of the patent instrument obfuscates the disclosure.[10] Legal doctrines in patent law that measure the sufficiency of disclosure are important to account for the information discontinuity.

Information is a substitute for learning effort. The patent-reading artisan was not part of the invention development team, but need not expend the effort under the design phase, because she can obtain the benefit of that effort from the disclosed design information.

Enablement, then, under the Norden model, focuses on the interrelationship between the design and prototype phases. If the effort to build the prototype is so great as to be undue, given design information disclosed in a patent, then the claim is not enabled. Norden modeled each phase of the research

---

6. Peter V. Norden, *Curve Fitting for a Model of Applied Research and Development Scheduling*, 2 IBM J. RES. & DEV. 232, 232, 233, 236 (1958) [hereinafter Norden, *Model*] (noting that the "study investigates broad, stable patterns and relationships in the R & D process, so as to provide a basis for at least limited improvement in forecasting accuracy").

7. *See* RICHARD E. FAIRLEY, SOFTWARE ENGINEERING CONCEPTS 79-80 (1985).

8. *Id.*

9. *See id.* at 79.

10. *See* Sean B. Seymore, *The Teaching Function of Patents*, 85 NOTRE DAME L. REV. 621, 632-41 (2010) [hereinafter Seymore, *Teaching Function*] (discussing the reasons patents instruments are written in "patentese"); *see also* Jeanne C. Fromer, *Patent Disclosure*, 94 IOWA L. REV. 539, 566-69 (2009); Gideon Parchomovsky & Michael Mattioli, *Partial Patents*, 111 COLUM. L. REV. 207, 209, 229-32 (2011).

and development project with an effort curve, where effort rose, peaked, and then fell.[11] The curves for the phases overlap.[12] Thus, a very high and wide prototype curve, representing great effort over a long time, suggests a claim that is not enabled because the design information was insufficient. Great effort, or quantitatively large effort, is not necessarily undue experimentation because that rubric includes qualitative assessments, but the quantitative aspects of enablement correspond with the Norden model.

The model also fits well with other patent law doctrines, most notably the best mode disclosure doctrine.[13] It also helps explain the doctrines that eliminate the patentee's disclosure burden for information related to manufacturing the invention at scale or efficiently. In Norden model terms, these doctrines say that information represented by the effort under the product phase is not information that patent law obligates for disclosure, unless the claims map to that space and/or a best mode is present in that space.

Part II reviews these insights from the Norden model generally. Part III brings these insights to the disclosure doctrines for software patents, with particular emphasis on the unpredictability factor for undue experimentation within enablement. The model corresponds well with enablement and best mode but does not correspond as well with other disclosure-prompting doctrines whose role is related to defining the claim. Thus, the review in Part III of written description, definiteness, and means-plus-function (§112 ¶ 6) claim limitations helps establish the contours of applicability for the Norden model.

The discussion of Part III also reviews the current state of the law for software patent disclosure: disclosure burdens are light and do not require disclosure of source code for the software. Thus, software patents may represent the high-water technology in patent law for having your cake and eating it too: trade secrecy protection attaches if the licensing and distribution of the software is according to proprietary licensing: distribution of object code, keeping source code secret. Within this review of software patent disclosure law, the Article contrasts the continuum of possible disclosure modes with the Norden model and patent law's current requirements.

Part IV then completes the article by arguing for a change to one of the requirements: reducing the categorical approach to unpredictability in the software arts. All of software should not be deemed predictable. Many niches are, but some are not.

Unpredictability is one of eight Wands factors that define undue experimentation,[14] but it is particularly important among the factors. Technologically, Part IV explains potential sources for unpredictable or unreliable behavior in software systems. Pragmatically, the progression of software tech-

---

11. FAIRLEY, *supra* note 7, at 79-80.

12. *Id.*

13. *See infra* notes 45, 230 and accompanying text (applying the model, respectively, to conception and obviousness).

14. *In re* Wands, 858 F.2d 731, 737 (Fed. Cir. 1988).

nology since the time of the precedent influencing enablement for software patents suggests a failure by the law to recognize the changes in the technology. Moreover, the disclosure doctrines in software patents have not responded to the expansion of patentable subject matter in the area of software patents. The discussion also helps show that patent law does not necessarily specify what it means by unpredictability, whether the unpredictable arts doctrine only attaches to ungovernable or inestimable items in nature or based on natural principles. Software is different as a discipline because it processes encoded information, where the encoding is derived from human thought. For some, this processing would not fit within a definition of what is "nature." Regardless, the Norden model suggests that an effort-based perspective on disclosure brings notions of unpredictability into the software arts in a nuanced and niche-specific manner.

II. EFFORT CURVES IN TECHNOLOGY DEVELOPMENT

That new technology develops in phases is familiar. The phases have common concepts: a design phase, a build phase, and perhaps a commercialization phase. Also familiar is that the phases overlap. Consider software development. Much of the public understands that a beta test version of a software product is somewhere between the build and commercialization phase. A variety of labels might apply for the phases of technology development.[15] This Part presents a model for the phases of effort within a research and development project. Although the model originates from a 1958 article,[16] the model remains influential in software engineering, particularly in the area of estimating the human effort and associated cost to develop software.[17]

A. The Norden Model

Peter V. Norden spent his career in technology management, much of it with IBM, and "for more than forty years he was involved in research and development management and design and development in manufacturing and related industries."[18] He studied the staffing patterns of research and devel-

15. *See* Ted Sichelman, *Commercializing Patents*, 62 STAN. L. REV. 341, 347-54 (2010) (discussing steps within the innovation process).

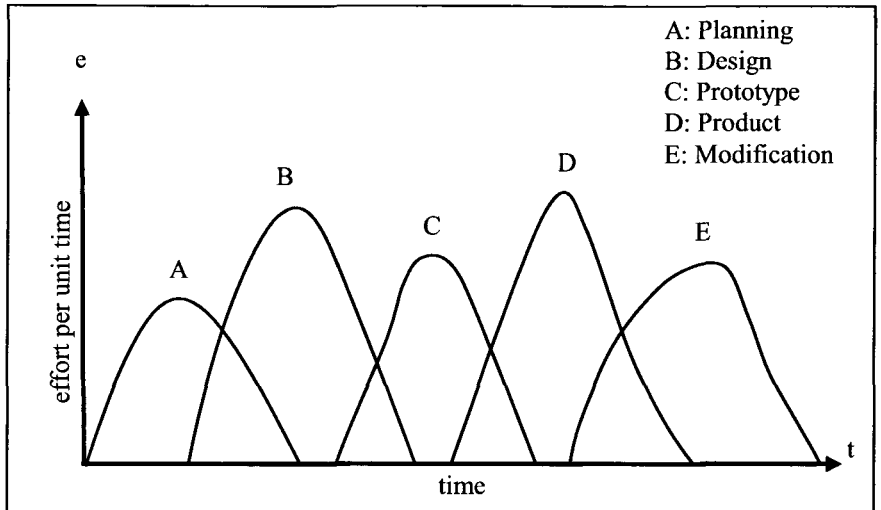
16. *See* Norden, *Model*, *supra* note 6.

17. FAIRLEY, *supra* note 7, at 79-80.

18. *Peter V. Norden*, INST. FOR OPERATIONS RES. AND THE MGMT. SCI. (INFORMS), http://www.informs.org/About-INFORMS/History-and-Traditions/Miser-Harris-Presidential-Portrait-Gallery/Peter-V.-Norden (last visited May 6, 2011).

opment (R & D) projects, and his work has been represented, in part, by Figure 1 below.[19]

Figure 1 – Norden Effort Curve Model for R & D Projects



Note: this representation has been redrawn by the author based on Figure 3.6 at page 80 of the FAIRLEY book, and that Figure 3.6 is itself an adaptation of the diagrams in the Norden model.

Norden considered curves A through E in aggregate to develop insights used later by software project managers to develop models for estimating the cost of developing software.[20] Those models underlie commercial software and information products available at the time of this writing and used by licensees to help estimate the cost for large software development efforts.[21]

Among the insights of Figure 1 is that effort rises, peaks, and falls over time for each phase of an R & D project. As modeled this way, the phases

19. FAIRLEY, *supra* note 7, at 79-80 (amalgamating several figures and depictions in the Norden article to arrive at the portrayal of Figure 3.6, pg. 80, with that figure depicted in the main text as redrawn by the author); Norden, *Model*, *supra* note 6, at 233, 236.

20. FAIRLEY, *supra* note 7, at 79-81. *See, e.g.*, F.N. Parr, *An Alternative to the Rayleigh Curve Model for Software Development Effort*, SE-6 IEEE TRANSACTIONS ON SOFTWARE ENG'G 291, 291 (1980), *available at* http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=01702731; *Timeline: A History of Industry-Leading Innovation*, QSM, http://www.qsm.com/about-us/timeline/ index.html (last visited May 6, 2011).

21. *See, e.g.*, QSM, http://www.qsm.com (last visited May 6, 2011); *The QSM Advantage*, QSM, http://www.qsm.com/about-us/index.html (last visited May 6, 2011).

overlap. As the next phase ramps up, the prior phase is winding down.[22] Effort can represent human time and energy, or its proxy, dollar cost to generate such. Although the Norden model has been influential within estimating methods for software project cost and staffing,[23] it is a general framework for consideration of R & D projects.

One outcome for some applied R & D is patents.[24] The purpose of this section is to introduce the Norden model in relation to inventive activity in which the inventor might apply for a patent. In this introduction, the focus is curves B, C, and D. The differentiation between curves A and B is not significant for this Article's application of the Norden model to patent disclosure doctrines, principally enablement. Nor is curve E relevant. But the relations among curves B, C, and D can be expressed in terms of patent law doctrines governing disclosure.

B. Prototype in Light of Design

Enablement in patent law relates curve B to curve C. That is, enablement relates the design, curve B, to the prototype, curve C. For validity, a patent claim must be enabled.[25] Enablement is a legal standard by which the claim is assessed against the disclosure in the patent instrument.[26] Enablement tests sufficiency of disclosure by this formulation: does the provided information enable an artisan in the field of the technology to make and use the claimed invention without undue experimentation.[27]

The enablement relation between curves B and C will be explored in greater depth in later parts from a doctrinal perspective. Thus, further detail about the enablement standard, and its judicially created undue experimenta-

---

22. Conceptually, it might be possible for two non-adjacent effort curves to overlap. For example, prototyping could start before planning is completed, meaning that curve C would start before curve A finishes.

23. FAIRLEY, *supra* note 7, at 79-80 (discussing Norden's research as a precursor to estimating software project cost based on staffing levels).

24. DAN L. BURK & MARK A. LEMLEY, THE PATENT CRISIS AND HOW THE COURTS CAN SOLVE IT 37-41 (2009) [hereinafter THE PATENT CRISIS]; Tallam I. Nguti, *Patent Law: Doctrinal Stability – A Research and Development Definition of Invention Is Key*, 20 VAL. U. L. REV. 653, 654, 659-60, 694 (1986).

25. 35 U.S.C. § 112 (2006).

26. *Id.* Atlas Powder Co. v. E. I. du Pont de Nemours & Co., 750 F.2d 1569, 1576-77 (Fed. Cir. 1984); Sean B. Seymore, *Heightened Enablement in the Unpredictable Arts*, 56 UCLA L. REV. 127, 130 (2008) [hereinafter Seymore, *Enablement*].

27. CFMT, Inc. v. YieldUp Int'l Corp., 349 F.3d 1333, 1338-40 (Fed. Cir. 2003); 3 DONALD S. CHISUM, CHISUM ON PATENTS, § 7.03[4] (2009); Fromer, *supra* note 10, at 546, 570.

tion proviso,[28] will come later. For present purposes, the focus is on the basic nature of the enablement relation between the design and prototype curves.

Information is often a substitute for effort. The Norden model represents an R & D project to develop new knowledge. Consider an expert group seeking to understand a completed R & D project, but who has not been exposed to the research. If the group was given the plan, it would not need to expend the effort under curve A to develop a plan. But, the expert group would need to do the design, expending the effort under curve B. Similarly, if given the design, it would not need to expend the effort under curve B. With the design in hand, the group could develop the prototype, but it would have to expend the effort under curve C to do so. This sequence would continue its relationship between adjacent curves moving through the phases of a project.

This framework applies to patent disclosure even though Norden studied teams of persons applied to an R & D project.[29] Patent law conceptualizes many of its doctrines from the perspective of a skilled artisan in the field. The statute in several locations articulates this perspective.[30] That Norden focuses on groups,[31] and patent law focuses on a hypothetical "reasonable person of patent law,"[32] does not undermine the framework of studying patent law disclosure doctrines through the Norden model. Actual inventors are sometimes joint.[33] Inventive activity, while sometimes solo, is often an organized enterprise activity. Patent law reserves inventorship for the human mind(s) originating the conception of a detailed inventive concept.[34] But those inventors often direct others in the effort to design and perhaps prototype a new invention.[35] The substitution of learning effort with information applies regardless whether a single person is learning the disclosed technology or a group of technologists is learning it.

The effort curve perspective on enabling disclosure assumes an equivalent effort scale for the original inventor as compared to the artisan attempting to make and use the disclosed invention. The original inventor's effort was to

---

28. Seymore, *Enablement*, *supra* note 26, at 147-150.

29. FAIRLEY, *supra* note 7, at 79-80.

30. *See* 35 U.S.C. § 112 ("person skilled in the art to which it pertains"); *id.* § 103(a) ("person having ordinary skill in the art to which said subject matter pertains").

31. Norden, *Model*, *supra* note 6, at 235.

32. Lance D. Reich, *One of Skill in the Art in Software Engineering: The Rising Tide*, 84 J. PAT. & TRADEMARK OFF. SOC'Y 269, 270 (2002); *see also* Timothy R. Holbrook, *Patents, Presumptions, and Public Notice*, 86 IND. L.J. 779, 779-81 (2011). Indeed, "the PHOSITA need not be a single person." Daralyn J. Durie & Mark A. Lemley, *A Realistic Approach to the Obviousness of Inventions*, 50 WM. & MARY L. REV. 989, 993 (2008).

33. 35 U.S.C. §§ 116, 262.

34. Oka v. Youssefyeh, 849 F.2d 581, 583 (Fed. Cir. 1988).

35. *See, e.g.*, *id.* at 583-84.

overcome a gap in her knowledge. The practicing artisan's effort also is to overcome a knowledge gap, but one that hopefully is narrowed by the patent disclosure.[36] As to measuring disclosure, patent law does not differentiate the "person having ordinary skill in the art" (PHOSITA) in those two scenarios.[37] This observation, for purposes of applying the Norden model, is sufficient to conclude that the scale for measuring effort, whatever it is (and it need not be

---

36. Indeed, the knowledge gap must be narrowed to some degree for enablement to be satisfied. *See* Fromer, *supra* note 10, at 571-73 (as part of an argument calling for revisions in the law governing patent disclosure, noting that patent documents are a mixture of technical and legal information, which, among other effects, has an obfuscating effect on beneficial absorption of the technical disclosure); Parchomovsky & Mattioli, *supra* note 10, at 209 (noting that comprehensive "disclosure can dramatically reduce research and development costs for other inventors"). In this context, one potential difference between an R & D group and an inventor is that "it is . . . not required that the inventor understand or disclose the principles that make the invention work, [so] long as it does work." ALAN L. DURHAM, PATENT LAW ESSENTIALS: A CONCISE GUIDE 71 (1999) (also noting that "[i]f the specification provides the information needed to make and use the invention, the patentee's theory as to why it works can be completely misguided.").

37. One might be inclined to conclude that an inventor is always a "person having ordinary skill in the art." While likely true most of the time, the statute does not require this status. *See* 35 U.S.C. § 101 ("*[w]hoever* invents") (emphasis added); *id.* § 102 ("[*a*] *person* shall be entitled to a patent unless") (emphasis added); *id.* § 103(a) ("Patentability shall not be negatived by the manner in which the invention was made.").

The section 103 formulation allows for surprise discoveries, which might be serendipitous and outside the inventor's technical expertise. *See* Sean B. Seymore, *Serendipity*, 88 N.C. L. REV. 185, 188-192 (2009) [hereinafter Seymore, *Serendipity*]. The courts have noted this approach.

> It should be clear that that hypothetical person is not the inventor, but an imaginary being possessing "ordinary skill in the art" created by Congress to provide a standard of patentability . . . . Realistically, courts never have judged patentability by what the real inventor/applicant/patentee could or would do. Real inventors, as a class, vary in their capacities from ignorant geniuses to Nobel laureates . . . .

Kimberly-Clark Corp. v. Johnson & Johnson, 745 F.2d 1437, 1454 (Fed. Cir. 1984) (emphasis removed).

In effect, even if an inventor is not a PHOSITA, the disclosure filed must meet sufficiency standards evaluated from the perspective of a PHOSITA. Sometime after invention, and somewhere along the way to the patent office, perhaps with the help of patent lawyers who oftentimes have technical knowledge, the inventive concepts are written down in words (and sometimes diagrams) such that there is disclosure to the public in the patent instrument. *See* Fromer, *supra* note 10, at 554-55, 564-69. The quality and sufficiency of that disclosure is judged from the perspective of a PHOSITA. *See* U.S. DEP'T OF COMMERCE, U.S. PATENT & TRADEMARK OFFICE, MANUAL OF PATENT EXAMINING PROCEDURE §§ 2161.01, 2164.06(c) (8th ed., 8th rev. July 2010) [hereinafter MPEP], *available at* http://www.uspto.gov/web/offices/pac/mpep/mpep.htm.

specified), is at least equivalent between the originator and later user of the knowledge.[38] Rough equivalency of scales means that the effort curve relations observed in R & D also speak to the relationships among the curves for the artisan making and using the disclosed invention.[39]

---

38. Picking some scale to measure effort and then measuring from that scale is different from the question whether the disclosure in curve B (design) is in a form to facilitate reduced effort by non-originators of the design information as such others attempt to make the prototype of curve C. The scale selection and proposition of equivalency of measuring effort derives from the proposition of rough equivalency in expertise of the readers, that is, the experts consuming and using the information. The dichotomy of originators versus later-in-time, non-originating users is where the patent instrument introduces a disconnect in the information, and patent law attempts to measure the degree of that disconnect via enablement. *See* Fromer, *supra* note 10, at 570.

With the original R & D group progressing through the curves, the information transference is likely more efficient than the patent scenario. The R & D group has continuity with documents and experimental data. They have information structure within their community of researchers, making a form of institutional memory. Thus, the R & D group's use of the information of curve B in its effort under curve C is likely an efficient transference of information from phase to phase (or at least more efficient than if another group starts cold to use design curve B to make a prototype involving effort of curve C).

The opposite is likely the case with a patent instrument posing as the information to substitute for effort under curve B. But that observation is just another way to make the point about why enablement can be modeled using the two effort curves. Patents systemically obfuscate technical disclosure for a variety of institutional reasons. *See* Seymore, *Teaching Function*, *supra* note 10, at 632-41 (discussing the reasons patents instruments are written in "patentese"); Keith E. Witek, *Developing a Comprehensive Software Claim Drafting Strategy for U.S. Software Patents*, 11 BERKELEY TECH. L.J. 363, 369 (1996) (presenting "three new types of software claims").

The more "patentese" and obfuscation, the more disclosure might be insufficient, and the more enablement will create invalidity risk for the claims. *See* Seymore, *Teaching Function*, *supra* note 10, at 634. An effort perspective on the obfuscation problem highlights the harm of the practice in ways complementary to other scholarship, noting that such practices are inconsistent with the norms of technical documentation in science and technology. *See generally* Sean B. Seymore, *Rethinking Novelty in Patent Law*, 60 DUKE L.J. 919, 928-29 (2011) (noting, like the other articles of Professor Seymore cited herein, that the *Rethinking Novelty* article is "part of a larger project to bridge the disconnect between patent law and the norms of science").

39. *See generally* Richard R. Nelson, *Intellectual Property Protection for Cumulative Systems Technology*, 94 COLUM. L. REV. 2674, 2675 (1994) ("Since software development does not differ significantly in this regard from much of industrial research and development, then if patents are appropriate for many of the modest routine advances of the latter, they may not be inappropriate on this count alone for protecting the former.").

C. Product in Light of Prototype

Information substituting for effort also is central to the relation between curve C, the prototype, and curve D, the product. If enablement is the patent law boundary doctrine between curve B and curve C, a corollary set of doctrines provide the boundary between curves C and D. Enablement creates an incentive for the patent applicant to disclose more information.[40] At some point, however, the applicant confidently can stop disclosing because certain types of information are not required for enabling disclosure.[41] For example, information needed to manufacture a claimed apparatus for market quantity is not disclosure required by patent law.[42]

Both patented inventions and R & D project output might stop short of product curve D. Many patents are not commercialized; patent law does not say that one must commercialize.[43] And the nature of R & D suggests that there are more attempts than commercialized outputs. Earlier, this Article mentioned that patent law requires enabling information to allow readers of the instrument to build a prototype. Restated using the Norden model, enablement states that the information you give in curve B has to be good enough to allow the artisan to build the prototype of curve C without undue experimentation, that is, without too much extra effort. While enablement puts a burden on the patent applicant, patent law helps the applicant as to curve D. She does not have to provide information to reduce the effort it might take another to create the product represented by curve D.[44]

---

40. *See* ROBERT PATRICK MERGES & JOHN FITZGERALD DUFFY, PATENT LAW AND POLICY: CASES AND MATERIALS 301-02 (3d ed. 2002); Yusing Ko, *An Economic Analysis of Biotechnology Patent Protection*, 102 YALE L.J. 777, 796-97 (1992).

41. ROBERT L. HARMON, PATENTS AND THE FEDERAL CIRCUIT 204-05 (5th ed. 2001) (noting that enabling information does not necessarily require disclosure of a scientific theory or principle about how or why the invention works).

42. Christianson v. Colt Indus. Operating Corp., 822 F.2d 1544, 1562 (Fed. Cir. 1987) ("Patents are not production documents, and nothing in the patent law requires that a patentee must disclose data on how to mass-produce the invented product . . . ."), *vacated on other grounds*, 486 U.S. 800 (1988).

43. *Id.* ("Many inventions are never manufactured; the decision to manufacture may be taken well after the patent has issued . . . ."). Commercialization of a patent could include making and selling products or licensing the patent to allow others to make and sell products that embody what the patent claims without infringement risk from the licensor's patent. *See* Ko, *supra* note 40, at 796.

44. This logic assumes that: (i) claim scope is fully encompassed by the prototype as an embodiment of the claim and that (ii) the best mode doctrine does not require disclosure by the inventor of information an artisan would need to learn to be effective with efforts under curve D in an attempt to practice the invention. This situation is likely the case for the second point due to the nature of the best mode doctrine, the way claims are drafted, and the doctrines removing from patent applicants the obligation to disclose manufacturing information. For more on the best mode doctrine, see *infra* subsection III.A.2.

The Norden model, with the insight that information sometimes substitutes for learning effort, illuminates the enablement doctrine in several ways. While curve A earlier was put aside, one might note that patent law doctrines of conception and inventorship could also be modeled using the relation between curves A and B.[45] This observation helps show the applicability of the model, which suggests itself as a lens for some parts of patent law given the common environment of patenting and R & D. Having covered this applicability in its fundamentals, the next Part will use the Norden model to focus on certain aspects of enablement's undue experimentation proviso.

III. PATENT DISCLOSURE & UNPREDICTABILITY

The Norden model is a staple concept in software engineering for cost/staff estimating.[46] With its lineage from software, it comes to patent law in this Article to help articulate certain propositions about disclosure for software patents. This Part narrows the use of the Norden model to enablement for software patents.[47] Along the way, the other disclosure doctrines

---

45. Patent law has a two-part conception doctrine: The first part of the doctrine's test relates to curve A, while the second part relates to curve B. The test has, first, a directing conception, and, second, a detailed means. Oka v. Youssefyeh, 849 F.2d 581, 583 (Fed. Cir. 1988).

> Conception may conveniently be considered as consisting of two parts. The first part is "the directing conception" and may be defined as the idea or conception that a certain desired result may be obtained by following a particular general plan. The directing conception is often referred to as the inventive concept, thought or idea. The second part of conception is "the selection of the means for effectively carrying out the directing conception."

Id. (internal citations omitted).

The directing conception is not sufficiently detailed to vest inventorship, and thus original ownership, to an inventor when she is competing with another inventor under the first-to-invent system used in the United States. Seymore, *Serendipity*, *supra* note 37, at 197. To win the first-to-invent race, the inventor might need to be the first to have possession of the detailed means in order to have "priority" over other inventors. *See id.* at 198-205 (discussing conception and a related doctrine, reduction to practice, and noting how these might impact priority of invention).

To map this doctrine to the Norden model, the directing conception is perhaps like the mere plan of curve A. Somewhere in the effort under curve B, the design, the detailed means for an operable invention will be found. Thus, the Norden model view of patent law's two-part conception test would articulate the situation as follows: the effort of curve A that produces only a directing conception is not sufficient to vest inventorship. Or, more simply, curve A does not equate to conception, but somewhere in the movement into curve B, conception is likely met.

46. *See* FAIRLEY, *supra* note 7, at 79-80.

47. There is perhaps a need to specify what is meant by "software patents" in this Article. Mostly, this Article defines "software patents" as method or system claims that are implemented predominantly with source code. In other words, software is

will be assessed in terms of the model, although their intrinsic correspondence with the model is less than that for enablement. In the final section of this Part, this Article expands on the fundamental treatment of Part II to further demonstrate how one might understand enablement, undue experimentation, and unpredictability, through the Norden model.

A. Disclosure and the Software Arts

While some observers have argued that patent law should deemphasize disclosure,[48] disclosure's two traditional salutary purposes are to put information into the public domain after a patent expires, and, even during the term of the patent, to help other users and developers find the technology and perhaps build on it.[49] Both purposes are part of the tradeoff behind the incentive effect hoped for with the patent system: exclusionary rights for a time but

essential to make and use the invention; that is, software is a predominant part of making and using. This approach is a broader definition than saying, for example, that software patents are any method claims implemented by software running in a computer. The statutory categories are often given in two classes: "process" or article of "manufacture" claims. 35 U.S.C. § 101 (2006). Patent lawyers write claims to a "system" when drafting claims for computer-implemented inventions. *See generally* John R. Allison, Abe Dunn & Ronald J. Mann, *Software Patents, Incumbents, and Entry*, 85 TEX. L. REV. 1579, 1593-97 (2007) (noting that information technology innovates and operates under the influences of the patent system); Julie E. Cohen & Mark A. Lemley, *Patent Scope and Innovation in the Software Industry*, 89 CAL. L. REV. 1, 8-11 (2001) (discussing the progression of increasingly flexible modes of claiming a software invention); Robert P. Merges, *Software and Patent Scope: A Report from the Middle Innings*, 85 TEX. L. REV. 1627, 1664, 1664-75 (noting a party with a patent with system claims and discussing a software patent infringement litigation concerning compression algorithms for graphical data); Robert E. Thomas, *Debugging Software Patents: Increasing Innovation and Reducing Uncertainty in the Judicial Reform of Software Patent Law*, 25 SANTA CLARA COMPUTER & HIGH TECH. L.J. 191, 192-93 & n.6 (2009) ("Business methods are part of the continuum that includes software and computer-implemented inventions, and the courts treat business method and software as interchangeable in determining patentability.").

48. Timothy R. Holbrook, *Possession in Patent Law*, 59 SMU L. REV. 123, 132-39 (2006). *But see* Fromer, *supra* note 10, at 543-45, 548-61.

49. Dan L. Burk & Mark A. Lemley, *Is Patent Law Technology Specific?*, 17 BERKELEY TECH. L.J. 1155, 1161 (2002) [hereinafter Burk & Lemley, *Technology Specific*]; Robin C. Feldman, *The Inventor's Contribution*, 2005 UCLA J.L. & TECH. 6, at ¶¶ 52-58 (2005) ("One way of thinking about the question of whether the invention works is whether we could achieve the results the inventor claims if we follow what the inventor tells us. This is another way of getting at the issue of whether the inventor has given us enough that we will grant a patent right.").

disclosure to the public as partial recompense, and, hopefully, greater incentive for inventors and firms to develop new technology.[50]

Commentators, including the author, support the proposition that disclosure for software patents is not of high quality, meaning that the information given often does not sufficiently help an artisan make and use the invention.[51] This intuition is in part a motivating force for this Article. The point is empirical, and if the greater point is whether software patents are of lesser quality on other metrics, there are contrarians among the commentators.[52] This Article's purpose is not to replay the debate over software patent disclosure. Rather the purpose is to show it in a new light for one fulcrum of one disclo-

---

50. Dan L. Burk & Mark A. Lemley, *Policy Levers in Patent Law*, 89 VA. L. REV. 1575, 1649 (2003) ("In return for a period of exclusive rights over an invention, the inventor must fully disclose the invention to the public.") [hereinafter Burk & Lemley, *Policy Levers*]. *But see* Alan Devlin, *The Misunderstood Function of Disclosure in Patent Law*, 23 HARV. J.L. & TECH. 401, 406 (2010) (arguing that, as a matter of policy, "disclosure cannot enjoy a status commensurate with the incentives to invent and commercialize").

51. JAMES BESSEN & MICHAEL J. MEURER, PATENT FAILURE: HOW JUDGES, BUREAUCRATS, AND LAWYERS PUT INNOVATORS AT RISK 210 (2008) ("[T]he disclosure requirement is rarely a sufficient ground upon which to invalidate software patents . . . ."); Burk & Lemley, *Policy Levers*, *supra* note 50, at 1691 ("[S]oftware patents present unique obstacles to consummation of the patent law's traditional rights-for-disclosure bargain with the public."); Burk & Lemley, *Technology Specific*, *supra* note 49, at 1164-65 ("Unfortunately, the Federal Circuit's peculiar direction in the software enablement cases has effectively nullified the disclosure requirement for software patents. And since source code is normally kept secret, software patentees generally disclose little or no detail about their programs to the public."); Devlin, *supra* note 50, at 445 ("Patents issued in the IT sector routinely fail to meet the literal requirements of § 112, an outcome that the Federal Circuit has oddly facilitated."); William F. Heinze, *A Risk-Balancing Approach to Best Mode Disclosure in Software Patent Applications*, 84 J. PAT. & TRADEMARK OFF. SOC'Y 40, 44 (2002) ("[The] client would like to disclose as little as possible about its software, and yet, [the patent attorney] know[s] this is just what the best mode and enablement requirements were intended to prevent."); Robert J. Tomkowicz, *Uneasy Fit - Software Patents and the Duty of Disclosure in Patent Law*, 25 CANADIAN INTELL. PROP. L. REV. 223 (2010), *available at* http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1681644 (discussing the "insufficiency of disclosing a computer program's functionality in patent applications for certain categories of software"); Michael J. Walsh, *The Disclosure Requirements of 35 U.S.C. 112 and Software-Related Patent Applications: Debugging the System*, 18 CONN. L. REV. 855, 871 (1986) ("The failure to require disclosure of computer program listings removes an essential aspect of software development from the section 112 analysis.").

52. John R. Allison & Ronald J. Mann, *The Disputed Quality of Software Patents*, 85 WASH. U. L. REV. 297, 299-304 (2007) (collecting authorities stating propositions against which the study is aimed and using an empirical analysis to argue that, along several metrics, patents obtained by software-producing companies are not necessarily different in quality and value as compared to patents obtained by less specialized firms).

sure doctrine: unpredictability as a factor for undue experimentation within enablement. In doing so, this section will explain the enablement and best mode doctrines because they best resonate with the Norden model. If unpredictability is to be understood in terms of effort in the Norden model, then unpredictability perhaps does not have a place in the other disclosure doctrines, particularly written description.

1. Enablement

The patent statute requires an applicant to provide in the "specification" portion of the application information about "the manner and process of making and using [the invention], in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same."[53]

While the enablement provision of § 112 ¶ 1 speaks of "the invention," patent law views the invention from the perspective of each and every claim in the patent.[54] Each claim marks out a class of things - class of machines, class of methods, family of compounds - that the patent covers.[55] Each claim stands alone as a sort of mini-patent, separately enforceable by an infringement action.[56] Each claim must meet various requirements for valid issuance

53. 35 U.S.C. § 112 (2006). *See also* MERGES & DUFFY, *supra* note 40, at 299-301 (noting that enablement must be established on the date the patent allocation is filed).

54. Procedurally, in the United States, patent rights must be initially obtained through proceedings before the United States Patent and Trademark Office (PTO). *See* 35 U.S.C. § 1(a). Substantively, an applicant for patent protection drafts customized exclusionary rights around the invention through what are called "claims" – numbered sentences at the end of the patent instrument. *See id.* § 112. *See, e.g.*, U.S. Patent No. 5,261,101 (filed Feb. 28, 1990). For example, a typical issued patent instrument might be about fifteen pages long, with perhaps the last three pages containing about twenty claims. The typical patent includes a number of claims of different scope, ranging from a broad class to an intermediate class to a narrow class. *See generally* JANICE M. MUELLER, PATENT LAW 82-86 (3d ed. 2009). That is because an applicant cannot know whether or not a broad class will inadvertently include something that appears in the prior art – such as in another patent, a technical article, or a product on the market. *See id.* Claim invalidity shows up in a significant proportion of patent lawsuits. Mark A. Lemley, *Rational Ignorance at the Patent Office*, 95 NW. U. L. REV. 1495, 1500-03 (2001); *see generally* PATSTATS, http://www.patstats.org/ (tracking approximately forty issues in patent litigation cases over ten years, about half of which are invalidity issues) (last visited Sept. 6, 2011). Such an inadvertent inclusion renders the broad claim invalid; hence the need for additional, narrower claims as a sort of "backup" coverage. MUELLER, *supra*, at 82-86.

55. *See* HARMON, *supra* note 41, at 216.

56. *Id.*

and ongoing validity, including the disclosure requirements such as enablement.[57]

Compare the statutory language quoted above from § 112 ¶ 1 to the enablement standard given at the end of the first paragraph of section II.B, which asks, "does the provided information enable an artisan in the field of the technology to make and use the claimed invention without undue experimentation." As stated earlier, the judiciary added one more aspect to enablement: undue experimentation.[58] This proviso embellishes the statutory language, which is interesting as a matter of patent law doctrine, but the proviso's venerable nature is not at issue in this article.[59] The proviso is current law under the Federal Circuit's cases, as is the way patent law defines undue experimentation.[60]

Conceptually, the mere existence of the undue experimentation proviso means that some experimentation is allowed. The disclosure need not be perfect, only good enough. By good enough, if a skilled artisan has to run experiments, or build alternative embodiments, or try different materials, none of these efforts automatically show that the disclosure does not enable the claim. First, the "skilled artisan" is a legal fiction. Second, this construct represents a standard level of knowledge and expertise within a field. Third, against this construct, the hypothesized effort that the skilled artisan must undertake is measured. If it measures as too much, then it is undue. But, how to measure?[61]

As an example, in a leading case, *Atlas Powder*, the § 112 ¶ 1 issue was whether the patent claims were enabled in a patent owned by Atlas Powder for an explosive mixture.[62] DuPont, the party arguing that the claims were not enabled, pointed to the possibility of thousands of mixtures that one might need to try.[63] Despite this argument, the appellate court affirmed that the claims were enabled.[64] The possibilities arose from combinations of various fuels, salts, and emulsifiers.[65] The disclosure gave candidate lists of exempla-

57. *Id.* at 216-17.

58. CRAIG ALLEN NARD, THE LAW OF PATENTS 67 (2008).

59. HARMON, *supra* note 41, at 203; MUELLER, *supra* note 54, at 110.

60. CRAIG ALLEN NARD & R. POLK WAGNER, PATENT LAW 54-56 (2008).

61. Professor Feldman proposes an approach that calibrates the amount of allowed experimentation within enablement by asking whether that experimentation effort is trying to reach technological concepts not yet knowable because the knowledge front has not yet advanced that far. Feldman, *supra* note 49, at ¶ 129 ("Experimentation so great as to almost rise to the level of after-developed technology would surely constitute experimentation that is undue.").

62. Atlas Powder Co. v. E.I. du Pont de Nemours & Co., 750 F.2d 1569, 1576-78 (Fed. Cir. 1984) (stating that "the amount of experimentation, however, must not be unduly extensive").

63. *Id.* at 1576.

64. *Id.* at 1576-77.

65. *Id.* at 1576.

ry items in each category.[66] Thus, the salts, fuels, and emulsifiers listed were not exhaustive and did not all work in any combination.[67] The court concluded that the skilled artisan would find it routine work in chemistry to determine how to make some operable combinations.[68] Enablement did not require operability of every combination made from the candidate lists of all possible salts, fuels, or emulsifiers.[69]

The possible mixtures in *Atlas Powder* illustrates a common situation in patent law for a claim: the genus/species problem.[70] Each mixture is a possible species. The class of all possible mixtures is the genus. In other words, a single patent claim might cover thousands of combinations, which in the case of *Atlas Powder* are particular emulsifications. Patent law vernacular also might call, as a synonym to "species," each combination an "embodiment."[71] An embodiment is something that fits within the language of the claim.[72] In some claims, all possible embodiments are fully functional.[73] In *Atlas Powder*, that scenario was not the case. Some mixtures of the salt, fuel, and emulsifier, along with the other substances called for by the claim, might not be a stable mixture,[74] meaning they might explode prematurely.

To illustrate with another example, hypothetically given, consider:

> Patent claims can be, and often are, generic. For example, a claim might (hypothetically) cover *every* mousetrap that included the combination of (1) a spring, (2) a latch and (3) a trigger to unhook the latch and release the spring when disturbed by a mouse. The specification might disclose in detail only *one* example of such a trap-with a particular kind of spring, latch and trigger. If the claim covers other versions of the trap not discussed at all, is the specification adequate to enable one skilled in the art to practice the claimed invention?[75]

In the mousetrap example, a court likely will find the claim enabled because simple mechanical devices lead many persons to conclude that the device could be built without undue experimentation. The other implication of the mousetrap example is not an issue for this Article: what should the claim

66. *Id.*

67. *Id.* at 1576-77.

68. *Id.* at 1577.

69. *Id.* at 1576-78.

70. ROGER E. SCHECHTER & JOHN R. THOMAS, PRINCIPLES OF PATENT LAW 80-81 (2004).

71. *See* J. Benjamin Bai, *Enablement Issues Concerning Aggressively Broad Generic Claims*, 7 NW. J. TECH. & INTELL. PROP. 1, ¶ 1 (2008).

72. *See id.*

73. *See id.* at ¶ 19.

74. *Atlas Powder*, 750 F.2d at 1577.

75. DURHAM, *supra* note 36, at 71.

cover? In other words, what is its scope when used against an accused defendant in patent litigation? This question points to another purpose of enablement: cabining claim coverage through invalidity via lack of enablement. If the claim is not valid because it is not enabled, then the claim provides no exclusionary right to limit others from practicing the claim.[76]

The genus/species issue in *Atlas Powder* turned in part on the court's estimation that the chemistry at issue was sufficiently predictable and in part on the finding that many embodiments would work even though some would not. The court explained as follows.

> Even if some of the claimed combinations were inoperative, the claims are not necessarily invalid. "It is not a function of the claims to specifically exclude . . . possible inoperative substances . . . ." Of course, if the number of inoperative combinations becomes significant, and in effect forces one of ordinary skill in the art to experiment unduly in order to practice the claimed invention, the claims might indeed be invalid. That, however, has not been shown to be the case here.[77]

The court also noted that even though internal records of Atlas Powder showed up to 40% of its experiments had problems, the court accepted the district court's conclusion that this finding was in "essence because they were not optimal under all conditions, but such optimality is not required."[78] Considering the quantity of inoperative embodiments within the genus/species inquiry in *Atlas Powder* was the way by which the court evaluated undue

---

76. HARMON, *supra* note 41, at 203-06. A more nuanced aspect of enablement cabining claim scope is when courts consider enablement concepts within the claim construction process. *See, e.g.*, Epistar Corp. v. Int'l Trade Comm'n, 566 F.3d 1321, 1336-37 (Fed. Cir. 2009); Phillips v. AWH Corp., 415 F.3d 1303, 1315-16, 1323-24 (Fed. Cir. 2005). The court in *Phillips* noted:

> One of the best ways to teach a person of ordinary skill in the art how to make and use the invention is to provide an example of how to practice the invention in a particular case. . . . In the end, there will still remain some cases in which it will be hard to determine whether a person of skill in the art would understand the embodiments to define the outer limits of the claim term or merely to be exemplary in nature.

Id. at 1323. *But see* Jeffrey A. Lefstin, *The Formal Structure of Patent Law and the Limits of Enablement*, 23 BERKELEY TECH. L.J. 1141, 1181-89 (2008) (arguing that enablement is not a beneficial doctrine to limit claim scope).

77. *Atlas Powder*, 750 F.2d at 1576-77 (internal citations omitted).

78. *Id.* at 1577 (internal citations omitted).

experimentation.[79] In patent law, the enablement issue often arises in that genus/species framework but can arise in other frameworks as well.[80]

As a matter of doctrine, undue experimentation is explained further by reference to the Wands factors.[81] Several of these factors dominate the analysis in *Atlas Powder*.[82] The Wands factors are in essence the way in which patent law measures whether the hypothesized effort that the skilled artisan must undertake is too much and thus undue. An important factor is unpredictability. Unpredictability is next reinterpreted from the perspective of the Norden model before concluding this section with a treatment of enablement for software patents.

Because enablement is the patent law doctrine relating curve B to curve C in the Norden model, Figure 2 below uses only those two curves in a side-by-side presentation. On the left side is the original relationship depicted in Figure 1 above. The right side alters curve C to depict lack of enablement.

Assume that the amount of effort depicted under curve C, the prototype, in the left side is sufficient to declare the claim enabled. Within that effort, whatever experimentation exists is not so much as to be undue. The curve on the right side represents greater effort. The right-side curve C models prototype-building effort so great in light of the disclosure of curve C that patent law would say the claim is not enabled.[83]

---

79. *See id.*

80. HARMON, *supra* note 41, at 203 ("Whether [there is] undue experimentation . . . is not a single, simple factual determination, but rather is a conclusion reached by weighing many factual considerations.").

81. F. SCOTT KIEFF ET AL., PRINCIPLES OF PATENT LAW: CASES AND MATERIALS 183 (4th ed. 2008). The factors are called "Wands Factors" after *In re Wands*:

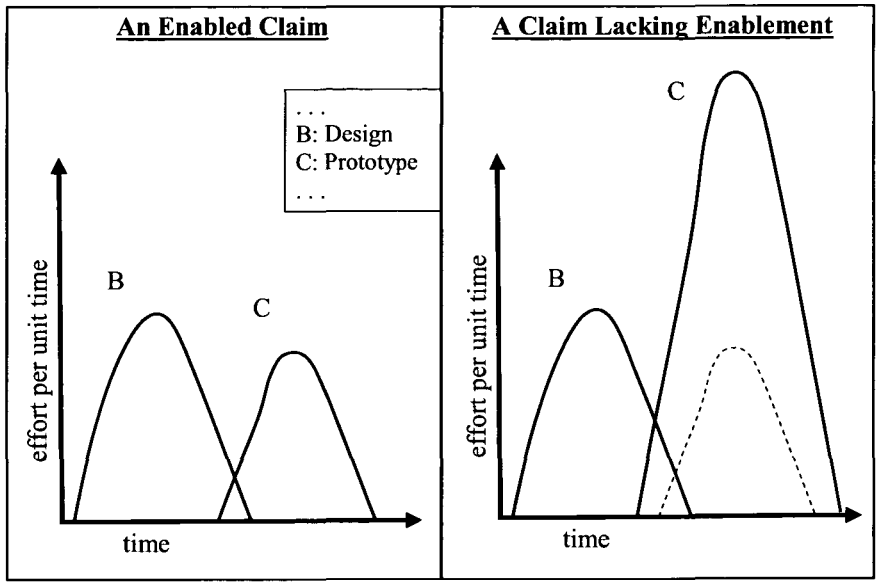
> (1) the quantity of experimentation necessary, (2) the amount of direction or guidance presented, (3) the presence or absence of working examples, (4) the nature of the invention, (5) the state of the prior art, (6) the relative skill of those in the art, (7) the predictability or unpredictability of the art, and (8) the breadth of the claims.

858 F.2d 731, 737 (Fed. Cir. 1988).

82. Among the Wands Factors discussed in the main text's short summary of the enablement issue in *Atlas Powder* are: quantity of experimentation, amount of guidance, working examples, and predictability. *See Atlas Powder*, 750 F.2d at 1576-78.

83. The presentation of Figure 2 increases the effort under curve C on the right side to illustrate lack of enablement, but the model could have instead reduced the disclosure of curve B, design information, to also make the same point. In this alternative illustration, on the right side curve C would remain the same size, but curve B would shrink. To demonstrate in a specific way, consider the hypothetical mousetrap example. Assume these facts: (i) the spring needs to deliver force in such a way as to cause motion of the trap within a range of 10 feet/second up to 50 feet/second; (ii) an artisan would find it really hard to select the right type of spring. If the design information does not give specifications for the force of the spring, or the derivative measure, the range of speed for the trap driven by the spring, the result is a shrunk curve B.

Figure 2 – Enablement and the Norden Model



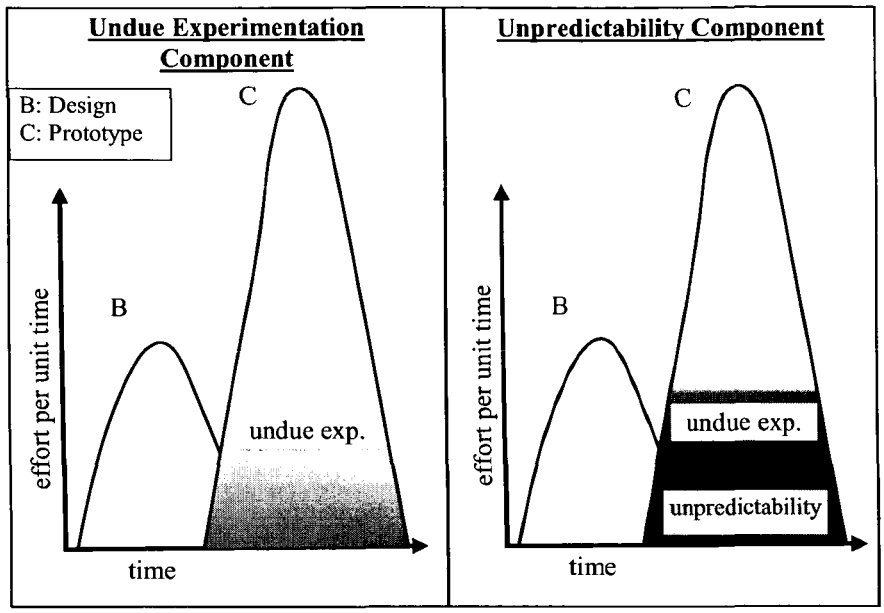
In characterizing the right side Norden model in Figure 2 above as effort "too great" such that there is lack of enablement, the approach should recognize that the test is not merely quantitative but has qualitative factors as an overall reasonableness measure for what is "undue."[84] In other words, the vertical effort scale should not be considered as an overly quantitative construct.

To expand the depiction, the next figure explores the possible components of effort attributable to elements of the relevant patent law doctrines under study: undue experimentation and unpredictability.

It results in lack of enablement under curve C because selection of a proper spring was hypothetically specified to be "really hard." Thus, this article's statement that "enablement in patent law relates curve B to curve C" is a result of the relative relationship between the two curves.

84. *See In re Wands*, 858 F.2d at 737; GREGORY A. STOBBS, SOFTWARE PATENTS 264-65 (2d ed. 2000).

Figure 3 – Enablement Effort Components Under the Norden Model



In Figure 3, the left side depiction shows some portion of the effort to build the prototype attributable to undue experimentation. That portion is shaded in gray in curve C. The right side adds to this the undue experimentation as attributable to unpredictability, shaded in black in curve C. The exact size and shape of curve C, as well as the quanta attributed to both undue experimentation and unpredictability, does not need to be specified with precision. Even without such precision, the Norden model effort curves help illustrate the doctrinal potency of undue experimentation and unpredictability in terms of effort for a follow-on artisan attempting to make and use the invention based on the patent disclosure.

The last topic for this section is to bring together the genus/species concept, the Norden model, and enablement for software patents. Many commentators believe that the genus/species approach to claim scope connects to chemical arts, but it is applicable to patent claims in general.[85] It is particularly applicable to software patents because they tend to claim technology with broad, multivariate language.[86]

---

85. Lefstin, *supra* note 76, at 1168-81.

86. BESSEN & MEURER, *supra* note 51, at 187, 194, 195-199 ("[O]n average, software patents suffer notice problems more acutely than patents drawn from most other areas of technology."). *But see* Andrew Chin, *On Abstraction and Equivalence in Software Patent Doctrine: A Response to Bessen, Meurer and Klemens*, 16 J. INTELL. PROP. L. 197, 200-201, 204-214 (2009). *See also* Supplementary

To use the example presented by BESSEN & MEURER,[87] consider U.S. Patent Number 5,758,257, filed Nov. 29, 1994 (the '257 patent), which describes itself as, "[a] system and method for scheduling the receipt of desired movies and other forms of data from a network which simultaneously distributes many sources of such data to many customers, as in a cable television system."[88] Claim 1 includes terms such as "customer profile" and "content profile."[89] These constructs are given in plain language, not any particular artisan language used among computer programmers.[90] The plain words result in an abstract claim, viewable as a genus sketched in words and containing many species depending on what information technologies (IT) are used to implement the scheduling system. The disclosure of the '257 patent lists several embodiments from an IT hardware perspective including One-Way Data Transmission System, Two-Way System, and Set Top Multimedia Terminal Embodiments.[91] In its text, the '257 patent uses the word "software" five times in relation to components of the claimed system, envisioning an implementation via software.[92] It never mentions a specific programming language to use to write the software.[93]

This last point connects to the current settled law for enablement with software patents for the most important, yet unimportant, issue. Source code

---

Examination Guidelines for Determining Compliance with 35 U.S.C. 112 and for Treatment of Related Issues in Patent Applications, Part 2, item II, 76 Fed. Reg. 7162, 7171 (Dep't of Commerce, Patent & Trademark Office Feb. 9, 2011) [hereinafter PTO Supplementary §112 Guidelines] *available at* http://edocket.access.gpo.gov/2011/pdf/2011-2841.pdf ("Although the specification need not teach what is well known in the art, applicant cannot rely on the knowledge of one skilled in the art to supply information that is required to enable the novel aspect of the claimed invention, when the enabling knowledge is in fact not known in the art.").

87. BESSEN & MEURER, *supra* note 51, at 196-97.

88. U.S. Patent No. 5,758,257 (filed Nov. 29, 1994).

89. *Id.*

90. *See id.*

91. *Id.* These three embodiments are species under a subgenus the patent labels as "Hardware Implementation of Profile System." *Id.* A different subgenus is labeled "Alternative Embodiments of Systems Which Use Agreement Matrix," and it contains these species: Video Distribution Systems; Video, Music and Bookstore Kiosks; and Data Retrieval Systems. *Id.* The last item listed, "Data Retrieval Systems" is, as a term within IT, itself so broad that it could be treated as a sub-subgenus. *Id.*

92. *Id.* In this discussion, I put aside the numerous questions (showing further species/embodiments) about the type of computer the software runs on, and whether it runs as compiled object code, or interpreted source code. The '257 patent envisions some embodiments running on set-top boxes for receiving cable television. *See id.* Around the date of issuance of the patent, May 26, 1998, these devices were perhaps not yet "computers" in the sense that many set-top boxes are at the time of this article. However, set-top boxes of that era typically ran compiled code, stored in firmware, with a processor specific to the task of receiving and displaying video.

93. *See id.*

need not be disclosed for a software invention.[94] Important because, if the answer were otherwise, the implications for software patents would be tremendous. Unimportant because the doctrinal answer is what it is, and, moreover, because patent law's disclosure doctrines exert no pressure toward anything close to source code disclosure.[95]

One commenter, Gregory Stobbs, describes software invention disclosure along a continuum:

> A proper specification lies somewhere on a spectrum between a black box and full source code. A black box is a descriptive concept engineers use when they want to simplify by hiding the details. The concept is simple: the black box is described simply as performing a given function, so that a given input subjected to that function, produces a given output. The details of how the function is performed are not disclosed. Software can be described like this.[96]

The "black box" idea may strike the uninitiated as no disclosure at all. However, if what is inside the "black box" is "conventional structure and can be determined without undue experimentation," then "[b]lock diagrams and functional descriptions are permissible" including use of "black boxes" to occupy the spots on the block diagram.[97] Stobbs also notes that sometimes source code, at the other end of the continuum, "may not provide a very good understanding of what the invention does."[98] Whether disclosure benefits arise from source code depends on a variety of factors, such as how well the programmers have commented the code, the programming language, and many other factors.[99]

---

94. HARMON, *supra* note 41, at 204; LEMLEY ET AL., SOFTWARE AND INTERNET LAW 198-99 (3rd ed. 2006) [hereinafter SOFTWARE & INTERNET LAW]; Michael Bondi, *Upholding the Disclosure Requirements of 35 U.S.C. § 112 Through the Submission of Flow Charts with Computer Patent Applications*, 5 SOFTWARE L.J. 635, 636 (1992) (arguing that, as compared to source code disclosure, "only disclosure by flow chart furthers the objective of the patent system"); Steven T. Naumann, *Compliance with 35 U.S.C. § 112 for Inventions Containing Computer Software: Is Disclosure of the Computer Code Required*, 4 SOFTWARE L.J. 443, 449-51 (1991).

95. *See* Amir A. Naini, *Convergent Technologies and Divergent Patent Validity Doctrines: Obviousness and Disclosure Analyses in Software and Biotechnology*, 86 J. PAT. & TRADEMARK OFF. SOC'Y 541, 557 (2004) (noting that for software patents the Federal Circuit has "established a low standard of disclosure for the purposes of satisfying the enablement and written description requirements.").

96. STOBBS, *supra* note 84, at 277.

97. *Id.* at 263.

98. *Id.* at 277.

99. *Id.* at 277-78. *See also* Tobi Carver Clinton, *Infringement and Software Claimed Under 35 U.S.C. § 112, ¶ 6: Software Function Is the Important Part*, 5 VA. J.L. & TECH. 4, at ¶¶ 44-48 (2000) (discussing varying degrees of detail for software

To further illustrate with the '257 patent, among the eleven figures in the patent, one figure is labeled "Software Block Diagram."[100] That figure, in its dazzling simplicity, is given in Figure 4 below.

Although the patent calls the figure a software block diagram, it provides more abstract information content than data flow diagrams, a more common and standardized software design approach.[101] However, methodologies for creating documentation before software source code is written vary across the information technology sector.[102] That the approaches vary significantly underscores the need to view software disclosure along a continuum.

Another point in the continuum is pseudocode. One definition for it is "simply source code, where the syntax rules are relaxed and where unimportant details are left out."[103] In comparison to the software block diagram in Figure 4 below, pseudocode disclosure would provide much more detail about how the software would implement the key step of the method of claim 1 of the '257 patent: "relating said selected customer profile with the content profiles for the data available from each data source to the customer at a particular time by determining a distance between a customer profile and a content profile in a characteristic space by calculating an agreement scalar for common characteristics."[104] The rest of that important step in the method is given in claim 1 by a formula to calculate the "distance" and "agreement scalar."[105]

---

disclosure); Jacqueline D. Lipton, *IP's Problem Child: Shifting the Paradigms for Software Protection*, 58 HASTINGS L.J. 205, 218-30 (2006) (describing aspects of computer programming, including modes of documentation, both generally and from a historical progression).

100. U.S. Patent No. 5,758,257 (filed Nov. 29, 1994).

101. STOBBS, *supra* note 84, at 323-29; *Chapter 9: Dataflow Diagrams*, STRUCTURED ANALYSIS WIKI, http://yourdon.com/strucanalysis/wiki/index.php?title=Chapter_9 (last visited May 18, 2011).

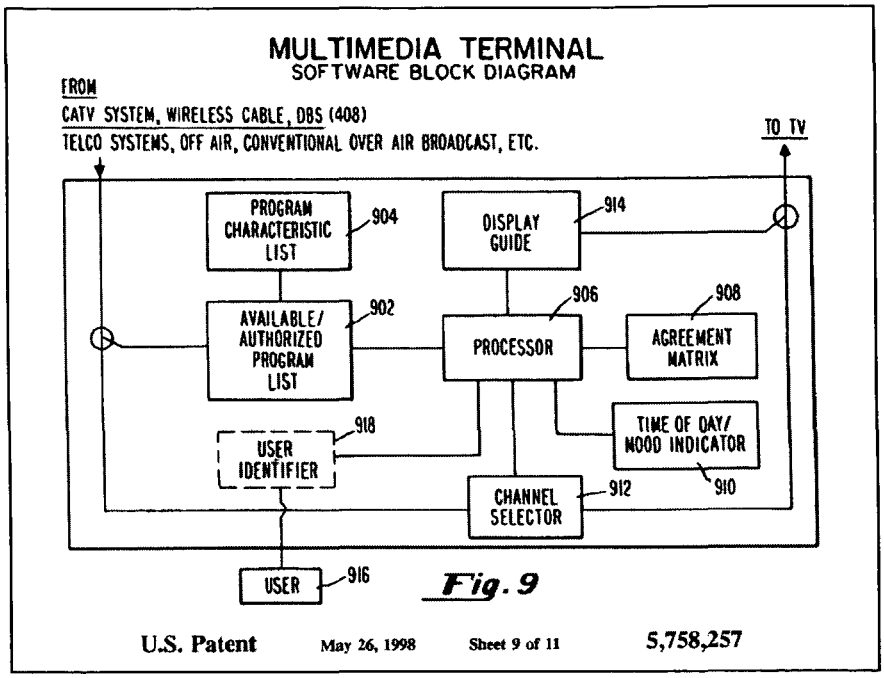
102. Daniel Jackson, *A Direct Path to Dependable Software*, COMM. ACM, Apr. 2009, at 84.

103. STOBBS, *supra* note 84, at 310-15.

104. U.S. Patent No. 5,758,257 (filed Nov. 29, 1994).

105. *Id.*

Figure 4 – Software Block Diagram Fig. 9 of the '257 patent – Method for Scheduling Access to Video Programs



With these illustrative examples and brief review of some spots on the continuum of disclosure for software as a backdrop, the law of enablement for software patents is a story about how disclosure typically is not required beyond the level of detail depicted in Figure 4 above, plus the associated textual descriptions that narrate the figures of the '257 patent.[106] The doctrinal influences prompting this state of affairs often include at least two related presuppositions. First, the PHOSITA in software is readily able to generate source code to underlie diagrams[107] like that in Figure 4 above. Second, software technology is a predictable field like most areas of mechanical and electrical engineering.[108] To the extent these two presuppositions are true, from a Norden model perspective, the learning effort under curve C, to make a prototype, is small enough to not include experimentation that is undue. To the extent these presuppositions are not always accurate approximations of

106. Burk & Lemley, *Policy Levers*, *supra* note 50, at 1653 ("In certain industries, such as software, the enablement requirement is easily satisfied and therefore plays virtually no role in limiting the scope of claims."); *see also* MPEP, *supra* note 37, § 2164.06(c).

107. *See* HARMON, *supra* note 41, at 210.

108. *See* Burk & Lemley, *Policy Levers*, *supra* note 50, at 1653-54.

reality, then enablement is not filtering out some software patent claims where the prototyping learning effort would be undue.[109]

This section will conclude with one exemplary case on each side of enablement. The general thrust of Federal Circuit law as to software patent enablement is well canvassed by other academics.[110] These two cases allow further illustration to view the inquiry through the Norden model.

In *Northern Telecom, Inc. v. Datapoint Corp.*,[111] the infringement defendant, Datapoint, succeeded before the district court in an extensive trial with a finding that certain claims were not enabled.[112] The Federal Circuit, however, reversed this determination.[113] While software was necessary to make and use the claimed invention, the claims covered a data entry terminal.[114] Datapoint, like most patent defendants, raised other points of invalidity before the district court, including obviousness.[115] The appellate opinion used Datapoint's obviousness experts against Datapoint on appeal for the enablement issue.[116] To support an obviousness argument, several of those experts spoke about how it would be straightforward to make the claimed invention.[117] The appellate court acknowledged that one Datapoint expert "testified that additional information such as detailed flow charts, block diagrams, or source code listings were necessary in order to avoid spending experimental time."[118] But the court put this argument aside, citing precedent to arrive at the counter logic that mere "description of such information may be adequate to a skilled programmer . . . ."[119] The case thus cast a shadow on enablement for software patents from the odd vantage of the inherent oppositeness of some of the invalidity arguments available to patent defendants. Most cases follow the approach in *Northern Telecom* by setting a low threshold for software patent disclosure,[120] which makes the next case somewhat of an outlier.

The patent at issue in *White Consolidated Industries, Inc. v. Vega Servo-Control Inc.*[121] covered software that controlled machines that drill and mill

109. *See* Burk & Lemley, *Technology Specific*, *supra* note 49, at 1162-67.

110. *See, e.g.*, *id.* at 1162-67.

111. 908 F.2d 931 (Fed. Cir. 1990).

112. *Id.* at 941-43 ("The cause was vigorously litigated, the trial taking seventy days over a six-month period. The district court issued extensive findings of fact and conclusions of law, in a 219 page opinion.").

113. *Id.* at 945.

114. *Id.* at 933.

115. *Id.* at 934.

116. *Id.* at 941-43.

117. *Id.* at 941-42.

118. *Id.* at 942.

119. *Id.* (internal citation omitted).

120. *See* NARD & WAGNER, *supra* note 60, at 55-56.

121. 713 F.2d 788 (Fed. Cir. 1983).

metal.[122] The district court found the claims invalid under enablement and best mode, but on appeal the Federal Circuit reviewed only the enablement issue.[123] While sporting technology from an almost ancient era of computing, the enablement issue raises modern questions about software components and interoperability. One part of the software needed to make and use White's invention was a translator.[124] A specific translator software component, called SPLIT, was mentioned for use by the patent's disclosure.[125] In finding lack of enablement, the Federal Circuit found that there was "insufficient evidence . . . from which to conclude that suitable substitutes for SPLIT were known and widely available."[126] White argued that two third-party translators would be interoperable and useable within the skill of an artisan programming the system, but the Federal Circuit disagreed stating that "there is no basis in the record for finding that a person skilled in the art on reading the specification would know that another single pass [translator] would be suitable."[127] In other words, whether interoperable substitute software components were within the knowledge of a PHOSITA in this type of software was not proven by White to avoid the negative consequences for it of its thin disclosure in the patent.

In *Northern Telecom*, brashly using obviousness testimony for a purpose for which it was not intended, the court imagines a strong PHOSITA.[128] This approach leads to a hypothetical prototype curve C of small size in the Norden model. In *White Consolidated*, the court concluded that a key software component is necessary and that artisan knowledge would not include knowledge of substitutes or the capability to create the translator anew without too much effort.[129] In Norden terms, in *White Consolidated* the learning effort curve C was enlarged beyond what is undue.

---

122. *Id.* at 789.

123. *Id.* at 788-90.

124. *Id.* at 789.

125. *Id.*

126. *Id.* at 790. The court was hesitant to find the claims enabled on the basis of the availability of SPLIT as a proprietary, closed-source-licensed, translator software product because at the time of patent filing, "SPLIT was a trade secret of Sundstrand, White's predecessor in interest, and was available only by purchase from Sundstrand." *Id.* at 789. The court's discussion suggests some need to perhaps modify the translator as a component of the invention for different modes of operating the invention, which might have led to the court's hesitancy to declare enablement met; closed-source-licensed software is typically not modifiable by the licensee. *See* T. Robert Rehm, Jr., *Navigating the Open Source Minefield: What's A Business to Do?*, 10 WAKE FOREST INTELL. PROP. L.J. 289, 299 (2010) (discussing the characteristics of closed source software); Greg R. Vetter, *The Collaborative Integrity of Open-Source Software*, 2004 UTAH L. REV. 563, 587-88.

127. *White Consol. Indus.*, 713 F.2d at 790.

128. N. Telecom, Inc. v. Datapoint Corp., 908 F.2d 931, 934-37 (Fed. Cir. 1990).

129. *White Consol. Indus.*, 713 F.2d at 791.

2. Best Mode

The outcome in *White Consolidated* was influenced by the finding of a best mode violation. The appellate court only evaluated enablement, but the district court also found a best mode violation.[130] A best mode violation occurs when the inventor had a subjective "best mode" of practicing the invention and does not disclose it.[131] The best mode is an embodiment of what is claimed that the inventor thinks is superior to other embodiments or ways of practicing the invention.[132] In *White Consolidated*, this mode was an embodiment of the system using the software component SPLIT as the translator.[133] Specifically, "the enabling requirement and the best mode requirement go hand-in-hand due to the proprietary nature of SPLIT and its importance as the only single-pass language known at the time to work . . . ."[134] The trade secret status of SPLIT, as software whose source code was held secret and licensed only to users as object code, was key to the best mode finding.[135] Oftentimes, this consideration is part of the policy justification for the best mode doctrine – to specifically disgorge trade secrets of the inventor that are essential to practicing the invention in relation to the scope of the claim.[136]

Like enablement, the obligation to disclose the best mode does not reach to production information or customer requirements.[137] However, the best mode requirement attaches to information that has a nexus with the claim under consideration.[138] Thus, if the claim is a method or process or some technique within manufacturing, the information surrounding that technique would need to be disclosed for enablement and for the best mode embodiment if the inventor had one in her mind.

Beyond these parallels, best mode is also a disclosure doctrine that can be modeled with Norden effort curves in ways both similar and different to the relation enablement provides between curves B and C, that is, between design and prototype.

---

130. White Consol. Indus., Inc. v. Vega Servo-Control, Inc., No. Civ. 79-70826, 1982 WL 63799, at *35-40 (E.D. Mich. July 8, 1982).

131. HARMON, *supra* note 41, at 207-16 (discussing the two components of a best mode inquiry: the subjective assessment whether the inventor knew of a best mode, and the objective evaluation of the disclosure to see if that mode was disclosed); MERGES & DUFFY, *supra* note 40, at 263.

132. HARMON, *supra* note 41, at 207.

133. *White Consol. Indus.*, 1982 WL 63799, at *3-5.

134. *Id.* at *36.

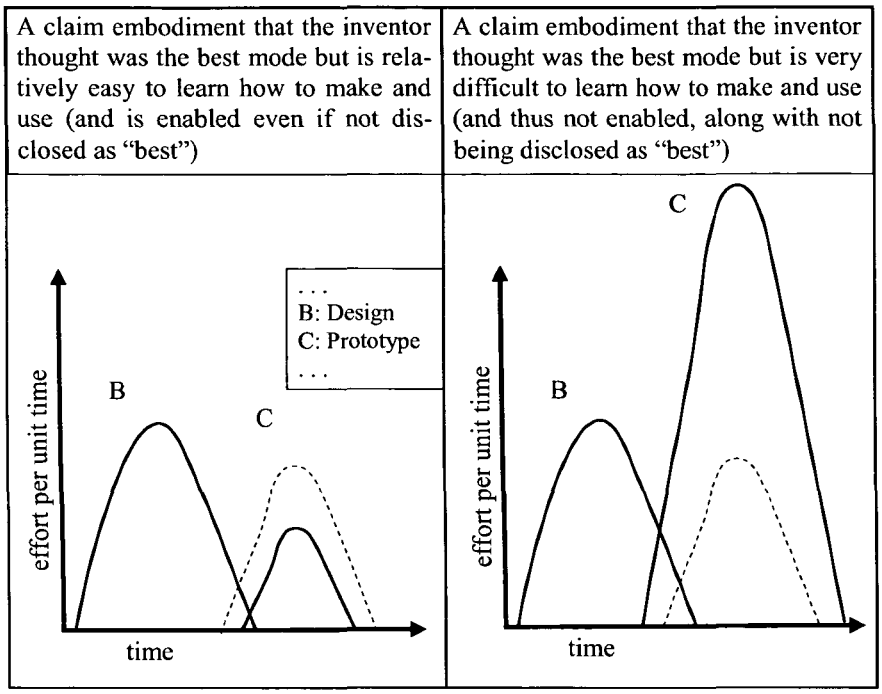
135. *See* White Consol. Indus., Inc. v. Vega Servo-Control, Inc., 713 F.2d 788, 790-91 (Fed. Cir. 1983).

136. *See id.* at 791.

137. HARMON, *supra* note 41, at 210.

138. *Id.* at 210-12.

Figure 5 – Best Mode and the Norden Model



The key differentiation between the left versus right curve sets in Figure 5 above arises from the insight that a best mode can be non-disclosed as to being "best," but not suffer from lack of enablement. Part of this insight is that the word "enabled," as a modifier, sometimes is applied to a claim and sometimes applied to specific embodiments or species fitting within what the claim covers. The disclosure might not provide enough information to allow the artisan to make and use a specific embodiment without undue experimentation, but it could enable other embodiments. A typical patent claim covers a class of things and thus is a genus of some sort. If a hypothetical narrow claim has only six embodiments, and all are enabled, patent doctrine also uses the modifier "enabled" to describe that the claim is not invalid for lack of enablement. Thus, the left side of Figure 5 above shows an enabled mode/embodiment that the inventor thought was best but was not disclosed as to having that status of "best." The right side shows an embodiment with that defect but also with the defect of requiring undue experimentation to discover how to make and use it.

Another inference flows from the idea that the enablement standard can be used to measure either a claim or an embodiment of a claim: best mode/embodiment disclosure violations oftentimes turn on something specif-

ic that was not disclosed, such as the hardness of a particular material.[139] Enablement for a claim, on the other hand, is a quantitative and qualitative assessment for the claim in full. The tendency for best mode violations to be for specific aspects of an embodiment has not translated for software patents into a requirement that source code be disclosed for a mode the inventor thought was "best."[140]

For example, in *Fonar Corp. v. General Electric Co.*, the Federal Circuit affirmed a jury's finding that no best mode violation occurred, meaning that the jury was convinced that the best mode was disclosed.[141] The patent covered medical imaging technology.[142] Some software routines were important in its use.[143] The court's holding places the needed disclosure far down the continuum of disclosure for software:

> As a general rule, where software constitutes part of a best mode of carrying out an invention, description of such a best mode is satisfied by a disclosure of the functions of the software. This is because, normally, writing code for such software is within the skill of the art, not requiring undue experimentation, *once its functions*

139. *See* Chemcast Corp. v. Arco Indus. Corp., 913 F.2d 923, 928-30 (Fed. Cir. 1990); MUELLER, *supra* note 54, at 113-14 (explaining that the best mode requirement "can be thought of as a sort of enablement-plus requirement"). That enablement applies to the claim or an embodiment of the claim suggests a similar insight with obviousness: that both the conception and reduced-to-practice embodiments should be the subject of inquiry. *See* Jeanne C. Fromer, *The Layers of Obviousness in Patent Law*, 22 HARV. J.L. & TECH. 75, 86 (2008) ("A layered understanding of invention suggests that in assessing an invention's obviousness, we ought to be concerned with the obviousness of both the conception and the reduction to practice, two quite different aspects of the invention.").

140. SOFTWARE & INTERNET LAW, *supra* note 94, at 204, 209; STOBBS, *supra* note 84, at 267-68; Heinze, *supra* note 51, at 43-44; Naumann, *supra* note 94, at 451-54. Naumann elaborates:

> Clearly, the Federal Circuit is willing to give inventors maximum flexibility when disclosing their inventions. The court considers programmers to be the functional equivalent of translators. The disclosure need only be sufficient so that a skilled programmer can bridge the gap and create a program. It appears that the court believes that a program listing is the equivalent of a production specification and thus will not require full disclosure of the source code. Because the PTO generally does not resolve the issue of best mode disclosure, challengers will be faced with proving a best mode violation by clear and convincing evidence. As long as the inventor discloses that a digital computer is contemplated as the best mode and discloses the information required by *Sherwood*, a challenger will find that invalidating a patent will be difficult.

Id. at 454 (emphasis omitted) (citations omitted).

141. 107 F.3d 1543, 1548 (Fed. Cir. 1997).

142. *Id.* at 1546.

143. *Id.* at 1546-48.

> *have been disclosed.* It is well established that what is within the skill of the art need not be disclosed to satisfy the best mode requirement as long as that mode is described. Stating the functions of the best mode software satisfies that description test. We have so held previously and we so hold today.[144]

Additionally, the court reproduced Figure 7 of the patent in the opinion in order to identify that figure as disclosure showing the essential componentry of the embodiment asserted by General Electric to be the non-disclosed best mode.[145] The referenced figure was no more complex than the "Software Block Diagram" depicted in Figure 4 above, but the Federal Circuit took it to "provide[] a description of the functions required" which was sufficient for effective disclosure of the best mode.[146]

Among the disclosure doctrines in § 112 ¶¶ 1-2, enablement and best mode correspond with the Norden model. This correspondence is because enablement and best mode measure sufficiency of disclosure for a later-in-time artisan seeking to practice the technology, at least as to building a prototype. For completeness, however, three other disclosure doctrines in § 112 are reviewed in the section below to relate their status for software patent disclosure.

3. Claim-Defining Disclosure Doctrines

Continuing the doctrinal theme for software patent disclosure, none of the three disclosure doctrines discussed below regularly require source code disclosure for software patent claims. Specific information elsewhere on the continuum of software disclosure suffices. Despite coming from three different paragraphs of § 112, the three doctrines share a purpose toward defining the claim.

a. Written Description

Until March 2010, when the Federal Circuit issued its opinion in *Ariad Pharmaceuticals, Inc. v. Eli Lilly & Co.*,[147] patent law was unsettled as to whether the written description requirement of § 112 ¶ 1 was separate from enablement.[148] The *Ariad* case clarified that a separate "possession test"

144. *Id.* at 1549 (emphasis added). *See also* Kenneth Canfield, *The Disclosure of Source Code in Software Patents: Should Software Patents Be Open Source*, 7 COLUM. SCI. & TECH. L. REV. 6, ¶ 35 (2006) ("A patentee therefore can generally meet the § 112 disclosure requirements without disclosing source code, with a possible exception if she contemplated a specific coding as a best mode.").

145. *Fonar Corp.*, 107 F.3d at 1549-50.

146. *Id.* at 1548.

147. 598 F.3d 1336 (Fed. Cir. 2010).

148. *See id.* at 1344, 1350-52.

arose from§ 112 ¶ 1, toward measuring whether, in the eyes of a PHOSITA, "the specification . . . describe[d] an invention understandable to that skilled artisan and show[ed] that the inventor actually invented the invention claimed."[149] Historically, written description was a doctrine applied only when claims were amended during prosecution, but *Ariad* certified a move in recent caselaw that it was a standalone doctrine, which was co-equal to enablement, even if aimed at different policy purposes.[150]

An important pre-*Ariad* written description software patent case is *LizardTech, Inc. v. Earth Resource Mapping, Inc.*[151] *LizardTech* is consistent with *Ariad*, and cited by the court in *Ariad*.[152] The Federal Circuit in *LizardTech* affirmed the district court's determination that claim 21 was invalid

---

149. *Id.* at 1351-54. The court explained further the role of the "possession test":

> "[P]ossession as shown in the disclosure" is a more complete formulation.
>
> . . .
>
> [T]he level of detail required to satisfy the written description requirement varies depending on the nature and scope of the claims and on the complexity and predictability of the relevant technology. For generic claims, we have set forth a number of factors for evaluating the adequacy of the disclosure, including "the existing knowledge in the particular field, the extent and content of the prior art, the maturity of the science or technology, [and] the predictability of the aspect at issue."
>
> . . .
>
> [W]e set out [no] bright-line rules governing, for example, the number of species that must be disclosed to describe a genus claim, as this number necessarily changes with each invention, and it changes with progress in a field. . . . And while the description requirement does not demand any particular form of disclosure, or that the specification recite the claimed invention in haec verba, a description that merely renders the invention obvious does not satisfy the requirement.

Id. at 1351-52 (quoting Capon v. Eshhar, 418 F. 3d 1349, 1357-59 (Fed. Cir. 2005)).

150. The historical role for written description relates to what is sometimes called "priority policing." MUELLER, *supra* note 54, at 122. This role relates to the ability of an applicant to modify both the claims and the specification during patent prosecution. *Id.* Modifications to the specification, however, are constrained to not introduce "new matter." 35 U.S.C. § 132 (2006). New or amended claims after filing must be supported under the written description doctrine by the originally filed disclosure. MUELLER, *supra* note 54, at 122-23. Any new words added to the specification after the original filing cannot be necessary to support any of the claims, unless those new words merely augment information already present. *Id.* at 121-23; *see also* Mark D. Janis, *On Courts Herding Cats: Contending with the "Written Description" Requirement (and Other Unruly Patent Disclosure Doctrines)*, 2 WASH. U. J.L. & POL'Y 55, 64 (2000) (discussing the relationship between written description and patent law's proscription against adding "new matter" in the disclosure to support a claim).

151. LizardTech, Inc. v. Earth Res. Mapping, Inc., 424 F.3d 1336 (Fed. Cir. 2005).

152. *Ariad*, 598 F.3d at 1341, 1353, 1358, 1362.

for failure to satisfy the written description requirement of § 112 ¶ 1.[153] The patent related to compressing image data, and the specification disclosed only one method for compressing the data in a particular way, by creating a "seamless DWT."[154] The genus/species problem arose in the context of the written description doctrine, "because there are no limitations in claim 21 as to how the seamless DWT is accomplished, claim 21 refers to [m]aking a seamless DWT generically."[155] The genus was claimed, but only one species within the genus was disclosed.[156] In this context with this technology, that limited disclosure was insufficient to show possession of the invention.[157] If the patent applicant had foreseen this outcome in litigation many years after filing, she perhaps would have included additional disclosure about additional ways to accomplish a seamless DWT.

Doctrinally, written description operates to give the patent applicant an incentive to give disclosure so an artisan can understand the boundary of patent law's exclusionary rights for the claim at issue.[158] To the extent written description measures sufficiency of disclosure, one way in which it is different from enablement is due to enablement's undue experimentation proviso.[159] Undue experimentation allows the applicant to come up short with the disclosure, and perhaps still have an enabled claim, provided that experimentation to close the gap is not undue. The *Ariad* factors for evaluating written description are highly contextual, but do not structurally influence the doctrine in the same way as undue experimentation is to enablement.[160]

This discussion leads to one negative insight from the Norden model for the written description doctrine as given by the *Ariad* factors: whether it is appropriate to include predictability of the technology as a factor when it is one of the Wands factors for undue experimentation.[161] In dissent in *Ariad*,

153. *LizardTech*, 424 F.3d at 1346-47.

154. *Id.* at 1336, 1337, 1344 (the term "DWT" stands for "discrete wavelet transform"). *See generally* Merges, *supra* note 47, at 1657-75 (reviewing the *LizardTech* case in detail as part of a discussion about the impact of patenting within the software industry).

155. *LizardTech*, 424 F.3d at 1344.

156. *Id.*

157. *Id.* at 1346.

158. Canfield, *supra* note 144, at ¶ 44.

159. *Id.* at ¶ 24 ("Enablement is a lesser requirement [than written description]; it deals not with what the inventor knows, but with whether he provides enough information for *another* to make and use the invention. If he provides descriptions close enough to code, it should be sufficient.") (emphasis in original).

160. *Ariad*, 598 F.3d 1336, 1344-52 (Fed. Cir. 2010) ("[A]lthough written description and enablement often rise and fall together, requiring a written description of the invention plays a vital role in curtailing claims that do not require undue experimentation to make and use, and thus satisfy enablement, but that have not been invented, and thus cannot be described.").

161. *Id.* at 1351 (the disclosure "required to satisfy the written description requirement varies depending on the nature and scope of the claims and on the

Judge Pauline Newman noted that this muddles the written description doctrine with enablement.[162] If undue experimentation is understood from a Norden perspective as additional learning effort that arises based on lack of predictability in the technology, this purpose does not apply in written description because the disclosure's purpose is not to provide information to make and use. The purpose is to provide information to allow the artisan to understand what was invented.

This written description critique arising from the Norden effort curve perspective is not the main point of presenting the Norden model. But the critique shows additional applicability of the model. To the extent the critique is without full force, a lesser claim by the critique is that predictability should operate differently within enablement versus written description. The more predictability is understood in terms of learning effort by a later-in-time artisan, the more that differentiation should strengthen, and the less applicable predictability is to written description.

b. Definiteness

As an artifact of § 112 ¶ 2, the definiteness requirement springs from the statutory language commanding that the applicant's filing have "claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention."[163] In one software patent case, the Federal Circuit found the claim term "aesthetically pleasing" for the user interface of kiosk software to be an indefinite claim limitation.[164] The claim term at issue went to user interface functionality, not an element of the software that would fall within the continuum of disclosure, including items such as data flow diagrams or pseudocode, much less source code.

Many of the definiteness cases for software patents are related to means-plus-function claims,[165] to be discussed immediately below. One ex-

---

complexity and predictability of the relevant technology") (citing Capon v. Eshhar, 418 F.3d 1349, 1357-58 (Fed. Cir. 2005)).

162. *Id.* at 1358 (Newman, J., additional views); *see also* Boston Scientific Corp. v. Johnson & Johnson, Nos. 2010-1230, 2010-1231, 2010-1233, 2010-1234, 2011 WL 2184283, at *14 (Fed. Cir. June 7, 2011) ("The majority's opinion further extends the written description requirement into the realm of enablement.") (Gajarsa, J., concurring).

163. 35 U.S.C. § 112 ¶ 2 (2006); *see generally* MUELLER, *supra* note 54, at 68-74.

164. Datamize, LLC v. Plumtree Software, Inc., 417 F.3d 1342, 1353-56 (Fed. Cir. 2005). The claimed software system allowed the user to arrange screen elements in a user interface; the software allowed the user to vary the screen elements, but the claimed system would keep the elements uniform and "aesthetically pleasing." *Id.* at 1349. The PTO guidelines characterize *Datamize* as a case about a subjective term without any information to limit it, quantify it, or bound it qualitatively. PTO Supplementary §112 Guidelines, *supra* note 86, at 7166.

165. David A. Kelly, *In the Wake of* Datamize *and* Halliburton: *The Recent Spate of Patent Invalidations for Indefiniteness and the Implications for Patent Holders*, 75

ample of a software patent definiteness issue that runs to the disclosure continuum is the term "sequence encoder" in a data transmission patent.[166] The court called "sequence encoder" a coined term, although the concept is understandable to an artisan.[167] The problem was that the term had no ordinary and customary meaning, nor was it expressly or inferentially defined.[168] A patent should clarify this type of term within the continuum of software disclosure to cabin it for a PHOSITA.

c. Means-Plus-Function (§112 ¶ 6) Claim Limitations

The last claim-defining doctrine is an artifact of § 112 ¶ 6: the means-plus-function claim limitation.[169] Here, patent law allows the applicant to claim structure in terms of the function it performs. The means-plus-function claim term is interpreted according to a statutory recipe that commands a search for the corresponding structure in the disclosure.[170]

BNA'S PAT., TRADEMARK & COPYRIGHT J. 456 (2008); *see generally* PTO Supplementary §112 Guidelines, *supra* note 86, at 7168 (discussing, as part of PTO examiner guidance for applying the definiteness doctrine, disclosure requirements for computer-implemented means-plus-function claim limitations).

166. Acacia Media Techs. Corp. v. New Destiny Internet Grp., 405 F. Supp. 2d 1127, 1133-39 (N.D. Cal. 2005).

167. *Id.* at 1133. *See generally* MUELLER, *supra* note 54, at 69 ("[I]t is possible to write definite claims to novel technology, even if existing words are inadequate, because the patent law permits the applicant to create new words with which to claim her invention.").

168. *Acacia*, 405 F. Supp. 2d at 1133.

169. 35 U.S.C. § 112 (2006). The term "limitation" is meant to indicate a part or portion of the language of a claim. The statute, however, at least in this location, uses the term "element":

> An element in a claim . . . may be expressed as a means . . . for performing a specified function without the recital of structure, material, or acts in support thereof, and such claim shall be construed to cover the corresponding structure, material, or acts described in the specification . . . .

Id. § 112 ¶ 6 is a claim construction technique commanded by the statute whenever one encounters a claim limitation that qualifies. The clearest signal that means-plus-function treatment applies is use of the word "means" in a claim, such as "means for encoding."

170. At first blush, without the statutory claim construction method commanded by § 112 ¶ 6, one might think that the claim term "means for encoding" is broader than "encoder." This impression is because on a plain meaning basis, the words "means for encoding" might capture more possible real-world structure than the alternative limitation of simply saying "encoder." Often, however, the claim construction method commanded by § 112 ¶ 6 reverses that initial intuition. This reversal is because the statute commands the reader to find the corresponding structure in the disclosure in the patent instrument. *See id.* That (often more specific) structure is the meaning of the limitation. For example, if the only embodiment disclosed in the specification for "means for encoding" is an encoding of information

For software patents, this search for structure sometimes has required greater specificity in the disclosure beyond what is represented by the example Software Block Diagram depicted in Figure 4 above,[171] but not specificity as to require source code.[172] For various reasons many software patents are written with many claim limitations that use means-plus-function treatment.[173] While this approach has led to some doctrinal questions about disclosure to support these claim elements, these concerns are outside the brief orientation needed here.

Along with written description and definiteness, means-plus-function claim treatment is a doctrine that measures whether a PHOSITA could understand the scope of the claimed invention.[174] The public should be on notice, or at least constructive notice, as to the boundary of the class defined by a claim.[175]

> Because claims delineate the patentee's right to exclude, the patent statute requires that the scope of the claims be sufficiently definite to inform the public of the bounds of the protected invention, i.e., what subject matter is covered by the exclusive rights of the patent. Otherwise, competitors cannot avoid infringement, defeating the public notice function of patent claims.[176]

This aspect of patent law doctrine, claim scope, does not resonate with the Norden model the way the model relates to enablement and best mode. Other than the three doctrines reviewed above, claim scope also involves claim construction, the process of interpreting the words of the claim, beyond that for means-plus-function limitations.[177] Claim interpretation is a precursor activity to evaluating enablement and best mode.[178] Thus, the doctrines

---

using Morse Code, then the claim term is narrow as compared to the myriad possibilities available under the plain meaning of the word "encoder" to an artisan.

171. *See* PTO Supplementary §112 Guidelines, *supra* note 86, at 67-68.

172. Generally, an algorithm must be disclosed as the corresponding structure for the software patent claim element. Aristocrat Techs. Austl. Pty Ltd. v. Int'l Game Tech., 521 F.3d 1328, 1333-38 (Fed. Cir. 2008) (corresponding structure need only be a disclosed algorithm). *See also* Thomas, *supra* note 47, at 233-37 ("For software, providing source code or detailed outlines of how the code performs its functions would satisfy the literal meaning of the enablement and best mode requirements. However, the CAFC has ruled that software claims do not have to provide this level of detail.").

173. SOFTWARE & INTERNET LAW, *supra* note 94, at 217-19.

174. *See* Halliburton Energy Servs., Inc. v. M-I LLC, 514 F.3d 1244, 1249 (Fed. Cir. 2008).

175. *Id.*

176. *Id.* (internal citations omitted).

177. Phillips v. AWH Corp., 415 F.3d 1303, 1312-14, 1323-25 (Fed. Cir. 2005) (en banc).

178. *See id.*; *see, e.g.*, SCHECHTER & THOMAS, *supra* note 70, at 187-88.

are not fully separable. But enablement is the primary focus here because it always applies, and it raises the issue of unpredictable technology.[179]

B. The Unpredictable Technology Doctrine

The claim that must be enabled is the claim as it is interpreted. Thus, as a precursor activity to enablement evaluation, claim construction influences the process. It also can work in reverse: the interpretative activity might skew in favor of validity. In other words, one canon of claim construction is to interpret the claim in order to save its validity.[180] However, this canon is not as potent as other canons,[181] of which patent law has several.[182] Fundamentally, a court's purpose in claim construction is to help establish the boundary of the exclusive right and to use that boundary to evaluate a number of invalidity doctrines, including novelty, obviousness, and disclosure doctrines such as enablement.[183] In the leading case on claim construction, the court notes that interpreting the claim to save validity is in part recognition that the PTO issued the patent after an administrative process to review the claims in order to issue them.[184]

While claim interpretation and claim scope will influence enablement, the reverse less frequently will occur.[185] After determining claim scope by

---

179. The best mode evaluation only applies when an inventor subjectively had a mode she considered "best." MUELLER, *supra* note 54, at 116.

180. *Phillips*, 415 F.3d at 1327 ("While we have acknowledged the maxim that claims should be construed to preserve their validity, we have not applied that principle broadly, and we have certainly not endorsed a regime in which validity analysis is a regular component of claim construction.").

181. *See id.*

182. SCHECHTER & THOMAS, *supra* note 70, at 301-04.

183. *See* Halliburton Energy Servs., Inc. v. M-I LLC, 514 F.3d 1244, 1249-50 (Fed. Cir. 2008).

184. *Phillips*, 415 F.3d at 1327 ("In . . . cases [where after applying all other claim construction tools the term is still ambiguous], we have looked to whether it is reasonable to infer that the PTO would not have issued an invalid patent, and that the ambiguity in the claim language should therefore be resolved in a manner that would preserve the patent's validity.").

185. When claim interpretation to save validity does occur, an example will show the mechanism. Assume the claim term "connection identifier encoder" in cellular telephone technology has ten recognized ways in which it can be implemented. The term is a genus with ten species. The disclosure gives only two ways to encode among the ten, and the other eight involve undue experimentation. The claim seems invalid for lack of enablement, unless a basis exists to limit the term to only the two disclosed embodiments through the process of claim construction. Assume for the two disclosed embodiments that the identifier is a number, but for the other eight the identifier is a 200 character text string. Finally, assume that disclosure makes this statement: "The encoding modes contemplated as the invention are only those that use a numeric identifier." There is now some chance for a claim interpretation that saves

interpreting terms needing construction, the analysis proceeds to evaluation of enablement with its undue experimentation proviso and unpredictability as a factor to understand what is undue.

Listed as one of the Wands factors, one might think that unpredictability would be contextual, but it seems to operate categorically. The courts and commenters articulate broad categories as unpredictable arts.[186] Whether it should be categorical is a critique that can be raised from a number of perspectives,[187] but here that perspective is the Norden model, taken up as to software disclosure in section IV.B below.

Under present doctrine, the more the technology is unpredictable, the more that experimentation might be undue given a level of disclosure in the patent. Expressed as such, the approach seems contextual with a sliding-scale. However, the proof process in patent litigation results in holdings that express categorically that (i) the claim covers mechanical engineering technology, (ii) mechanical arts are predictable, and thus (iii) a lesser level of disclosure can be tolerated to find the claim enabled.[188] Expert witness testimony for disclosure doctrines such as enablement is highly technical for many patents, and this testimony oftentimes leads to expert equipoise from the perspective of the court.[189] In expert equipoise, default rubrics like "electrical circuits are predictable" facilitate the judicial analysis.[190]

The effort perspective demonstrated in Figure 3 above isolates effort attributable to unpredictability in making a prototype, but the isolation is artificial to make the point that some degree of the effort can be attributable to that factor among the Wands factors. The artificialness is because the Wands factors interact among themselves by their very nature, but in that interaction unpredictability stands somewhat apart.[191] The more unpredictability is given

---

the validity of the claim: the fact that it lacks enablement is a possible factor in a court accepting the narrowing interpretation based on the numeric identifier statement. *See generally* NARD & WAGNER, *supra* note 60, at 56-58.

186. MUELLER, *supra* note 54, at 105 ("[O]ne Wands factor very often central to the inquiry is whether the invention is considered to be within a 'predictable' or 'unpredictable' technology."); Seymore, *Enablement*, *supra* note 26, at 136 ("Historically, the judiciary has not required a specific and detailed teaching for inventions in applied technologies like electrical and mechanical engineering because they are rooted in well-defined, predictable factors.").

187. DURHAM, *supra* note 36, at 71-72.

188. *Id.*; Seymore, *Enablement*, *supra* note 26, at 136-37.

189. *See* Naini, *supra* note 95, at 566 (noting the "battle of the experts" in patent litigation); Seymore, *Enablement*, *supra* note 26, at 149 n.122.

190. *See* Spectra-Physics, Inc. v. Coherent, Inc., 827 F.2d 1524, 1533 (Fed. Cir. 1987); *but see* Allen Organ Co. v. Kimball Int'l, Inc., 839 F.2d 1556, 1565-67 (Fed. Cir. 1988) (jury verdict finding lack of enablement upheld for electric circuit).

191. *See supra* note 81 (listing the Wands factors).

Consider the following discussion for how the Wands factors work among themselves from a Norden model perspective. Factors (2) and (3) can be aggregated: the amount of guidance, and working examples [hereinafter, aFactor23, for

in broad categorical sweeps, the more it truncates the inquiry as to whether experimentation is undue.[192]

The challenges in adjudicating patent infringement actions are numerous. Thus, resorting to categorical approaches is understandable, even if non-optimal.[193] The unpredictable technology shortcut also shows up in obviousness.[194] Its appearance there, like in enablement, is related to the PHOSITA.

---

aggregated Factor23]. aFactor23 is subsumed in Norden model curve B, the design information.

Similarly, factors (5) and (6) can be aggregated: state of the prior art, and skill of artisans [hereinafter, aFactor56]. aFactor56 does not rest with any particular Norden curve, but underlies them all since the effort is being made by a PHOSITA in all curves.

Continuing, factors (4) and (8) can be aggregated: nature of the invention and breadth of the claims [hereinafter aFactor48]. Like unpredictability (the 7th factor), aFactor48 is subsumed into Norden curve C, effort to build the prototype.

What remains is the first factor, quantity of experimentation, at which the "undue experimentation" proviso partly aims. In other words, the first factor can be understood as a hypothetical estimate of the situation at the time of filing: was that quantity undue with PHOSITA knowledge then? Evaluating that question through the Norden model places aFactor23 in the design curve, and places aFactor48 and unpredictability in the prototype learning effort curve. The work under each curve is done by a baseline artisan of aFactor56.

192. Consider this thought-experiment about the truncating effect of a categorical approach to unpredictability. The first approach is the ESTdiscipline approach, where the term "ESTdiscipline" stands for any degree commonly granted by most U.S. universities somewhere within engineering, science and technology. Assume that there are about four dozen such degrees, including, for example "Electrical Engineering" or "Network Communications." (Please forget, for this thought-experiment, that patents issue for areas outside the domains of ESTdiscipline.) The second approach is to use the U.S. PTO class system, of which there are about 550. *Classes Arranged in Alphabetical Order*, U.S. PAT. & TRADEMARK OFFICE (Dec. 2010), http://www.uspto.gov/patents/resources/classification/caa.pdf. The more granular taxonomy, the PTO class system, will have greater sensitivity to predictability in a particular niche area of technology. The focus on a more granular approach will suggest to the decisionmaking process, particularly before judges, to not use broad categorical assessments. The thought-experiment concludes with a real fact: the PTO's model for generating the taxonomy that is its class system is an approach that itself considers complexity of the technology, which is a factor that is often, under systems theory, linked to predictability. *Appendix A – Structure of the Modern Schedule*, U.S. PAT. & TRADEMARK OFFICE, http://www.uspto.gov/patents/resources/classification/handbook/appxa.jsp (last visited May 24, 2011).

193. Arti K. Rai, *Engaging Facts and Policy: A Multi-Institutional Approach to Patent System Reform*, 103 COLUM. L. REV. 1035, 1040 (2003) ("[T]rial judges, and the juries empanelled by trial judges, may be overwhelmed by the technology involved in patent cases.").

194. KSR Int'l Co. v. Teleflex, Inc., 550 U.S. 398, 421 (2007) (predictable solutions in the prior art weight in favor of a finding of obviousness).

Professor Seymore contrasts the enablement PHOSITA with the obviousness PHOSITA: the former has been a plodder, but after *KSR Int'l Co. v. Teleflex Inc.*, the obviousness PHOSITA is more potent.[195] He posits that "[p]ossibly as a result of KSR, however, the Federal Circuit has started to police enablement more carefully in the predictable arts."[196] Thus, Professor Seymore's analysis proposes an effect of the unpredictable technology doctrine in a different doctrinal area than the obviousness doctrine at the center of *KSR*. To the extent that assessment of unpredictability is nuanced to a subfield within a technology and sensitive to advancement of knowledge within that subfield, the rubric makes better sense for either enablement or obviousness. To the extent it ossifies doctrine to facts from a technological past that are no longer salient, the doctrine may be less than fully beneficial.[197]

IV. AN UNPREDICTABILITY DOCTRINE FOR THE SOFTWARE ARTS?

At each phase of the Norden model, the sources of unpredictability will vary, but tie to common issues of information deficit such as insufficient data sets or theoretical models, or lack of both, such that reliable extrapolations cannot be made; or tools and techniques that are themselves of only partial and perhaps intermittent efficacy.[198] Enablement, from the Norden model perspective, is making and using a prototype from the design information. Understanding the unpredictability influence on the undue experimentation part of enablement in terms of the Norden model suggests a more flexible approach reducing categorical conclusions. The flexible approach is apropos to a technological area as broad as one labeled "software" or "information technology." The expansive breadth of either term as a category belies characterizing the entire field as unpredictable or not in a binary fashion. This conclusion is doubly true given the quick rate of technological advancement in information technology since the inception of the electronic computer. Professor Durham also speaks to these themes:

---

195. Seymore, *Enablement*, *supra* note 26, at 132-36.

196. *Id.* at 137. *See also* Lefstin, *supra* note 76, at 1175-81 (noting a recent doctrinal shift in some situations where the Federal Circuit has required "full scope" enablement, and discussing conceptual issues with that nascent doctrine). The *KSR* PHOSITA conception has influenced the law of definiteness for software patents. *See* AllVoice Computing PLC v. Nuance Commc'ns, Inc., 504 F.3d 1236, 1242, 1245 (Fed. Cir. 2007) ("In software cases, therefore, algorithms in the specification need only disclose adequate defining structure to render the bounds of the claim understandable to one of ordinary skill in the art.").

197. Feldman, *supra* note 49, at ¶¶ 58-68, 104.

198. The information deficit issues will likely vary in character under each curve in the Norden model. Thus, the sources of unpredictability will vary from curve to curve.

> This distinction between the "predictable" and "unpredictable" arts strikes some as unsatisfactory. If a claim in the mechanical arts can be considered enabled even though it encompasses embodiments that are "inadequately disclosed" in the specification, of what use is the "predictability"' of the art? Is it true that mechanical contrivances are invariably more "predictable" than chemistry? Where do new arts, such as that of computer programming, fall? However, at least until the Federal Circuit or the Supreme Court decides to reexamine this issue, broad claims will be more readily tolerated in the mechanical and electrical arts than in the arts of chemistry and biotechnology.[199]

A. Unpredictability and Software

While the courts have analogized software, computer programming, and other aspects of information technology as similar in predictability to the discipline of creating circuits in electrical engineering; the software industry, or at least its academy, sometimes takes the opposite view.[200]

A theme for "software engineering," as a discipline, is to make software development more like engineering.[201] The conventional wisdom is that software projects, compared to common activities within electrical or mechanical engineering, frequently have greater risk of cost overruns and failure to implement all designed functionality.[202] Translating this project risk to

---

199. DURHAM, *supra* note 36, at 72 (internal citation omitted).

200. Jackson, *supra* note 102, at 78, 80-87.

> Perhaps in the future we will know enough about software-development practices that the very use of a particular technique will constitute evidence of the resulting software's quality. Today, however, we are far from that goal.
>
> . . .
>
> Actually, where data is collected, it is often suppressed; many companies withhold even basic information about the number and severity of defects in their products, even when issuing patches that purport to resolve them.
>
> . . .
>
> [C]ertified systems sometimes fail catastrophically.
>
> . . .
>
> Contrary to the intuition of many programmers, finding bugs should not increase confidence that fewer bugs remain; indeed, it is evidence that there are more bugs to be found.

Id. at 79-80. *See also* Jim Humelsine, Letter to the Editor, *Software Still as Much an Art as Science*, COMM. ACM, Jan. 2010, at 7.

201. James Larus & Galen Hunt, *The Singularity System*, COMM. ACS, Aug. 2010, at 72 ("The Singularity Project at Microsoft Research began by asking what modern operating-system and application software would look like if it were designed with modern software-engineering practices and tools.").

202. Jackson, *supra* note 102, at 79-81.

Norden model terms, the information generated in the design and prototyping phases does not allow for product implementation according to what is called for in the design. This dynamic also is present between the design and prototype phases because an insufficient design can impede successful prototyping. As conventional wisdom, this "not-quite-engineering" aspect of software development is changing and applies to varying degrees depending on the niche within information technology for which the code is developed.[203] There are niches where developers make software as reliably as some activities in other engineering disciplines. There are software niches where the opposite is true.

The "not-quite-engineering" phenomenon has had an impact on implementing software projects. These implementations may express as software products or as software a company runs internally.[204] Both of these are mostly effort under curve D, the product curve, in the Norden model. As such, unpredictability that influences this effort is not related to enablement as a matter of patent law doctrine.

However, to the extent delays, cost overruns, or failures to implement functionality, that occur under product curve D, are related to a poor prototype, the analysis encroaches on enablement. To the extent these prototype-building problems are related to unpredictable aspects of some niche ecology within information technology, the undue experimentation proviso of enablement can be implicated.

This section will end with a brief review of some of the potential generators of unpredictability within software ecologies. The presentation is general and means to emphasize the heterogeneous nature of the information technology ecology. The sources generating unpredictability could influence prototyping as well as product implementation. The prototype/product dichotomy is important to keep in view because it delineates where enablement applies.[205] The more these sources of unpredictability mostly or exclusively

---

203. Michael J. Lutz & Donald Bagert, *Software Engineering Curriculum Development*, IEEE SOFTWARE, Nov.-Dec. 2006, at 16, 16-17. Some aspects of software engineering or computer science (not necessarily fully overlapping fields) are highly mathematical, especially in the areas where software languages and tools are designed for others to use to build software. *See* Dennis de Champeaux, *Software Engineering Considered Harmful*, COMM. ACS, Nov. 2002, at 102, 102-03. *See also* Andrew Chin, *Computational Complexity and the Scope of Software Patents*, 39 JURIMETRICS J. 17, 20-24 (1998) (proposing standards by which algorithm analysis for a claimed method would impact the claim's infringement potency).

204. *See* Jay P. Kesan & Rajiv C. Shah, *Deconstructing Code*, 6 YALE J.L. & TECH. 277, 277 (2004) (discussing software development within various organizations).

205. *See* HARMON, *supra* note 41, at 199-200 (explaining that patents need not be production documents to be enabled). Sources of unpredictability that arise under the product curve in the Norden model are not relevant to enablement if they do not also arise under the prototype curve. For example, the common problem of malware attacks on computers and computer software would likely be an issue that falls under

attach to the product implementation phase, the less they implicate undue experimentation for a software patent. That dynamic works in reverse. To catalog these sources, a loose taxonomy will help order the presentation: layers and components.

Software runs on a hardware layer, with increasingly gray area at the interface, and software is typically multi-layered. The hardware layer can be a source of unpredictability in making and using a software invention, but this result has become less likely as technology progressed over the last several decades. A software layer might be made of various components. In choosing the term component, the goal is to encompass a variety of software mechanisms, such as software objects, classes of objects, modules, languages, messaging mechanisms, procedures, subroutines, and other similar items.[206] In other words, component is used in this article as an umbrella phrase without the assertion that it has that umbrella meaning throughout all of information technology.

If a software layer is made of components, then interactions between layers, or among the components in a layer, as well as fragility within any particular component, are potential sources of unpredictable behavior. One of the most important insights for software unpredictability from the layers taxonomy is that almost all software runs on top of other software. The only software that does not is software that runs on whatever stands for "hardware" within a particular niche within computing.[207]

At some level of abstraction, all software responds to and manipulates encoded information. Therefore, the idiom "lost in translation" can apply between components. A decent analogy is the meaning transference, or lack thereof, of two persons conversing, where one speaks the English language and the other speaks French. The interactions between these two persons are influenced in part by their ability to transfer meaning when neither speaks the other's language, or each only speaks the other's language poorly. Similarly software components can interact perfectly, well, or not at all, due to the in-

---

the product curve in the Norden model (unless the claims were for an anti-malware invention). As such, the malware problem, to the extent it introduces unpredictability into a software ecology, would be unlikely to be the type of unpredictability to count for undue experimentation. *See generally* Richard Warner & Robert H. Sloan, *Vulnerable Software: Product-Risk Norms and the Problem of Unauthorized Access* (Mar. 7, 2011), *available at* http://ssrn.com/abstract=1780280 (discussing ways in which software has vulnerabilities and relating those risks to tort law concepts).

206. Lipton, *supra* note 99, at 224-28 (discussing some of the components in the list).

207. Many software failures or unreliable behavior are often traceable to hardware failures or incorrect inputs, a concept within computing often called "garbage in, garbage out." *See generally* Frances E. Zollers et al., *No More Soft Landings for Software: Liability for Defects in an Industry that Has Come of Age*, 21 SANTA CLARA COMPUTER & HIGH TECH. L.J. 745, 746-67 (2005).

formation encodings they are designed to recognize and how well the components recognize them.[208]

Classically, the first layer above the hardware is thought to be the operating system.[209] A computer operating system, such as Micosoft Windows or the Linux kernel, is a collection of components.[210] Any of these components might themselves be fragile, to the point that building a rickety prototype depending on that component is an exercise in frustration for a programmer. Alternatively, a prototype might need to use several components in the operating system, each of which is robust, but whose interactions when used together are fragile, leading to unreliable behavior by the prototype attempting to use the several components.

Noticing the layers and components in software ecologies is to also notice that it is complex.[211] This observation is not meant to imply that it is more or less complex than other fields of technology but only to note that complex systems have inherent propensities for operational difficulties to arise from interactions among components in the system. Such difficulties might or might not reach to undue experimentation within enablement when a software patent is involved, but patent law should not sweep them from the inquiry.

B. Enablement

The enablement implications for these observations about generators of unpredictable behavior in software depend on two primary considerations. First, whether the unpredictability attaches to the prototype phase or to the product phase. Second, whether the purpose of the unpredictability doctrine is to account for things in nature, to account for things in human-constructed technological infrastructure, or to accommodate both.

To focus on the second consideration, a computer operating system is a technological infrastructure to manipulate encoded information. To the extent it does not work perfectly, it is a failure of human design and implementation, not a failure in understanding how some mechanism within nature works. An exception to this assertion is when computer failures arise from the underlying hardware due to malfunctions in that hardware from influences not understood within electrical engineering.[212]

---

208. *See* Larus & Hunt, *supra* note 201, at 76-78.

209. *See* Vetter, *supra* note 126, at 578-81.

210. *See id.* at 569 n.12.

211. Sallie Henry & Calvin Selig, *Predicting Source-Code Complexity at the Design Stage*, IEEE SOFTWARE, Mar. 1990, at 36, 36-37 (discussing software complexity metrics).

212. A famous hardware problem in the history of computing was Intel's faulty Pentium processor in the middle 1990s. *See generally* Tim Coe, et al., *Computational Aspects of the Pentium Affair*, 2 IEEE COMPUTATIONAL SCI. & ENGINEERING 18 (1995).

Putting that narrow exception aside, a different way to pose the second consideration is to remove it from software. When the tools needed to make and use what a patent claim recites, as well as the elements of the claim, are human-made items that are fragile, unreliable, and unpredictable, alone or in combination, does the unpredictability doctrine still attach? The answer seems to be "yes." For example, in *Atlas Powder Co. v. E.I. du Pont de Nemours & Co.*, some elements of the claim are items straight from nature, such as water.[213] But others are human-made ingredients.[214] The way the ingredients interacted was a function of natural laws, but chemistry provided a model[215] to give some predictability about how to influence those interactions. Thus, the enablement proposition in *Atlas Powder* seems to consider potential unpredictability from both nature and human-made technological infrastructure (certain ingredients). On the other hand, the human-made items in *Atlas Powder* depend on natural principles for their operation even if not existing in nature. This situation is different from a software invention that operates within encoding constructs derived from human thought.[216]

While a Norden model effort curve perspective suggests a non-categorical approach to unpredictability within undue experimentation, another commenter places this suggestion within a proposed alternative framework for the PHOSITA:

> To be useful to the PHOSITA determination, the fact-finder should not simply categorize whole disciplines as predictable or unpredictable. Courts should recognize that the level of predictability in an art can change over time. A fact-finder should examine the predictability of the particular field of invention on a case-by-case basis.[217]

---

213. 750 F.2d 1569, 1572 (Fed. Cir. 1984).

214. *Id.* at 1571-72.

215. The model was a "basic principle of emulsion chemistry" called "Bancroft's Rule." *Id.* at 1576.

216. Another possible approach to unpredictability that would exclude software unpredictability that is not hardware derived is to say that unpredictability only applies from one or both of these aspects: nature itself, or human-made tools relying on natural phenomena. This approach is perhaps suggested by the doctrinal origins of the unpredictable arts in fields such as chemistry. This approach, however, is narrower than that suggested by the Norden model, which views unpredictability as something that increases learning effort. Also, at some level, this discussion becomes metaphysical as to what definition one takes for "nature."

217. Joseph P. Meara, *Just Who Is the Person Having Ordinary Skill in the Art? Patent Law's Mysterious Personage*, 77 WASH. L. REV. 267, 290-94 (2002) (proposing a new standard for the court to use to understand how a PHOSITA thinks about a technology problem, including predictability of the art as a factor, toward emphasizing problem solving ability rather than credentials).

This call for a fine-grained approach might have benefitted patent law related to software if it had been heeded. By the time of Mr. Meara's article with the above-given quote, *State Street Bank & Trust Co. v. Signature Financial Group, Inc.* was more than four years in the past.[218] Thus, *State Street*'s influence of increasing the domain of patents over information technology was being felt doctrinally and with an upswing in the number of software patent applications.[219]

The problem, however, is that patent law made no corresponding reassessment of the disclosure doctrines for software patents, even with a specialized appellate court as the primary influence over patent law. After *State Street*, not only were there more software patents, but they carried broader claims and reached to more areas of human endeavor.[220] The broader the claim, the greater the potential genus defined by the claim. This breadth leads to greater potential for lack of enablement of the genus. But the continuing meme analogizing software to electrical circuits or other electrical engineering concepts dampened the more nuanced response suggested by Mr. Meara's article.

Recalibrating the enablement disclosure doctrine in light of increased software patent claim breadth would involve recognizing several related considerations. First, that claims evolved to increasingly broad scope for software patents. Second, that software, software ecologies, and information technology became more complex, both within a single computer and among computers as networking became ubiquitous in wired and wireless forms. Third, that programming and software development had changed since the time of invention for technologies such as those involved in *Northern Telecom Inc. v. Datapoint Corp.*

The temporal dimension of the third consideration is stark. The filing date of the patent at issue in *Northern Telecom* is in July 1971, forty years past the time of writing this article.[221] As recently as 2003, a Federal Circuit judge cited *Northern Telecom* for the proposition that source code disclosure

218. 149 F.3d 1368 (Fed Cir. 1998), *abrogated by In re* Bilski, 545 F.3d 943 (Fed. Cir. 2008) (en banc), *aff'd sub nom.* Bilski v. Kappos, 130 S. Ct. 3218 (2010). There was at one point in patent law an exception to eligible subject matter called "business method[s]," where the claims covered items such as an accounting method or perhaps a method of demonstrating a product or a method of compensating a manager. *Id.* at 1375. However, this exception no longer exists. *Id.*; *see also* Bilski v. Kappos, 130 S. Ct. 3218 (2010).

219. Michael J. Meurer, *Business Method Patents and Patent Floods*, 8 WASH. U. J.L. & POL'Y 309, 310-13, 320-24 (2002).

220. Cohen & Lemley, *supra*, note 47, at 9-14 (discussing how, even before the *State Street* case, software became easier to claim due to decisions by the U.S. Patent and Trademark Office).

221. U.S. Patent No. 3,760,375 (filed July 26, 1971).

was not required for software patents.[222] This pattern shows that disclosure doctrines, particularly enablement, are tied to a technological past that cannot reflect the current realities of developing software nor reflect the complex ecology that is computing. The *Northern Telecom* opinion recognizes that disclosure will "vary according to the nature of the invention, the role of the program in carrying it out, and the complexity of the contemplated programming, all from the viewpoint of the skilled programmer."[223] However, the court stated with respect to the technology of that case that "[t]he claimed invention of the '375 patent is not in the details of the program writing . . . experts for both sides testified that an experienced programmer could, without unreasonable effort, write a program to carry out the invention of the '375 patent."[224]

The software for the claimed device in *Northern Telecom* was more akin to electrical engineering concepts than many of the software patent claims flooding into the U.S. patent system after *State Street*. To further emphasize this temporal incongruence, consider the actual device depicted in the '375 patent, given in Figure 6 below.

222. Moba, B.V. v. Diamond Automation, Inc., 325 F.3d 1306, 1325 (Fed. Cir. 2003) (Rader, J., concurring):

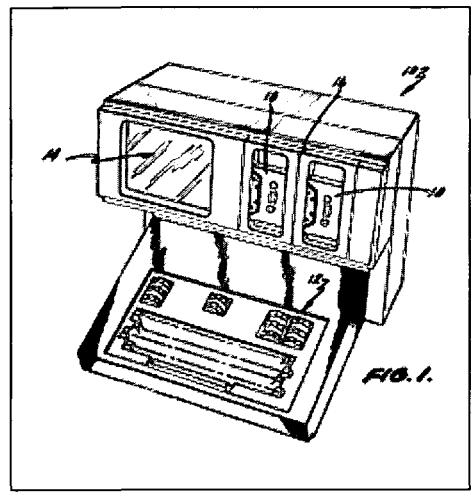
> This [*Lilly* doctrine] burdensome disclosure standard is tantamount to requiring disclosure, for a new software invention, of the entire source code, symbol by symbol, including all source code permutations that would not alter the function of the software. Ironically, the Federal Circuit has expressly rejected such a requirement for software inventions, but apparently enforces the requirement for biotechnology.

Id.

223. N. Telecom, Inc. v. Datapoint Corp., 908 F.2d 931, 941 (Fed. Cir. 1990).

224. *Id.*

Figure 6 – Fig. 1 of the '375 patent – Data Entry Terminal



The ancientness of the device in Figure 6 above demonstrates the staleness of disclosure doctrines as applied to software patents. Is this technology really an appropriate baseline analogy to use for claims of the sort reviewed in relation to Figure 4 above, discussing the '257 patent for a "method for scheduling the receipt of desired movies"?[225]

Particularly as to enablement, undue experimentation, and unpredictability, the considerations that the *Northern Telecom* opinion outlines, that disclosure requirements should vary among software patents, should be reinvigorated. The Norden model effort perspective amplifies this suggestion. One might ask why enablement and undue experimentation went stale as a doctrine while patent law was under the purview of a specialized court of appeals and while computer technology was advancing by leaps and bounds. Perhaps patent law should not expect that the specialization in the court would extend to specialization in following the technical disciplines that fit within the domain of patents. Perhaps litigants rarely emphasize undue experimentation in challenging enablement given the strong early tone of the law in cases such as *Northern Telecom*. Perhaps the difficulty of making an enablement case while also arguing obviousness as a patent litigation defendant influences litigants to emphasize the obviousness issue, where hindsight might help influence the factfinder in favor of the defendant seeking invalidity of the claim.[226] Perhaps the experts willing to be involved in the cases are not prone

225. U.S. Patent No. 5,758,257 (filed Nov. 29, 1994).

226. *See generally* Gregory N. Mandel, *Another Missed Opportunity: The Supreme Court's Failure to Define Nonobviousness or Combat Hindsight Bias in* KSR v. Teleflex, 12 LEWIS & CLARK L. REV. 323 (2008) (discussing non-obviousness and hindsight within that doctrine). Hindsight could also have an impact with unpredictability because enablement is evaluated as of the time the patent claim is

to see unpredictability in the world of software despite academic literature suggesting otherwise. Finally, perhaps most unpredictability is found in making a product, not a prototype, and thus does not apply as a matter of enablement doctrine.

The court in *Northern Telecom*[227] approvingly quotes a passage from *In re Sherwood* that recognizes, in its first sentence, that software disclosure and software programming exist along a continuum.[228] But the second sentence of the quoted passage equates programming to creating a "mathematical methodology to bridge the gap between the [input and output]."[229] This conception might belong in a 1970s sense of software development but not to present times. The reference to "mathematical methodology" suggests a false analogy to more predictable information technology niches, such as many areas of electrical engineering.

The unpredictable technology doctrine perhaps was never meant to operate categorically. Its usefulness as a judicial rubric for close cases with expert equipoise seems understandable but not optimal for software given the growth and change in it as a technology and the growth of software patents as a domain within patent law. While one disadvantage of the judicial process is that courts might miss such developments, an advantage of a specialized appellate court is that when they are noticed change can occur more rapidly. Into this fray, this Article suggests resort to the perspectives of the Norden model to understand unpredictability and undue experimentation from the perspective of learning effort. This approach dampens the easy slide into categorical perspectives for unpredictability.

Beyond enablement, unpredictability can influence written description and obviousness. The Norden model perspective does not suggest using unpredictability within written description as a claim-defining doctrine. While unpredictability has a place in the doctrinal structure of obviousness, understanding this fit is not the focus of this Article, and the fit is imperfect from

---

filed. *See* HARMON, *supra* note 41, at 197. As the enablement and unpredictability assessment moves further away from the time of filing, hindsight would likely lead a factfinder to see less unpredictability, particularly if the technology matures and strengthens.

227. *N. Telecom*, 908 F.2d at 941 (quoting *In re Sherwood*, 613 F.2d 809, 816-17 (C.C.P.A. 1980)).

228. *In re* Sherwood, 613 F.2d at 816-17. The full passage used by the court in *Northern Telecom* from *In re Sherwood* is as follows:

> In general, writing a computer program may be a task requiring the most sublime of the inventive faculty or it may require only the droning use of clerical skill. The difference between the two extremes lies in the creation of mathematical methodology to bridge the gap between the information one starts with ("the input") and the information that is desired ("the output").

N. Telecom, 908 F.2d at 941 (quoting *In re Sherwood*, 613 F.2d at 816-17).

229. *N. Telecom*, 908 F.2d at 941 (quoting *In re Sherwood*, 613 F.2d at 816-17).

the Norden learning effort perspective.[230] But making and using without undue experimentation, as the core of enablement, resonates with the learning effort perspective and signals a need for reconsideration of the unpredictable technology doctrine in the software arts.

V. CONCLUSION

Disclosure is an important part of the policy trade-off to grant patent rights. To be meaningful, disclosure requires legal standards to measure when it is sufficient. Enablement measures disclosure sufficiency, but its undue experimentation proviso allows disclosure to be less than perfect. Whether a technology niche is unpredictable is an important consideration in whether experimentation is undue. All of these concepts are understood from the perspective of the Norden model, which expressed phases of research and development as effort curves over time. For example, enablement, in the Norden model, measures whether information generated in the design phase and disclosed in a patent is sufficient to build a prototype without undue effort. With that approach and the insight that information can substitute for learning effort, the Norden model suggests new perspectives for enablement and other doctrines in patent law. The model arose from the study of research and development, it is a staple in software as a management and estimating theory, and this Article applies it specifically to software patents. Thus, the general insights from the model as to enablement are acute for software because the trend has been to lump software into a macro-characterization as a predictable technology. Changes over the last several decades in software patent eligibility, claim scope, and software technology belie that approach. The learning effort perspective suggests that unpredictability should be a fine-grained approach, both generally and particularly as to software technology. Under a flexible approach to unpredictability, courts hopefully will apply the doctrine to more areas within the software and information technology ecologies. The great breadth of these technologies and their rapid development suggest that there are niches or subfields where patent law should give unpredictability greater consideration than the doctrine has generally allowed.

230. An attempt to map obviousness to the Norden model would need to equate the design information to the prior art, and the invention to the prototype. However, the correspondence is less than enablement because there is some sense that obviousness involves more than mere learning effort. Unpredictability, however, has multiple places where it might influence the obviousness inquiry: assessing the level of skill of the artisan; the baseline question – is the claim as a whole obvious?; in conjunction with the reasonable expectation of success component of the now disfavored teaching, suggestion and motivation (TSM) test; and as a secondary or objective consideration.