Michigan Law Review

Volume 113 | Issue 5

2015

Installation Failure: How the Predominant Purpose Test Has Perpetuated Software's Uncertain Legal Status Under the Uniform Commercial Code

Spencer Gottlieb
University of Michigan Law School

NOTE

Installation Failure: How the Predominant Purpose Test Has Perpetuated Software's Uncertain Legal Status Under the Uniform Commercial Code

*Spencer Gottlieb**

Courts have struggled to uniformly classify software as a good or a service and have consequently failed to apply a consistent body of law in that domain. Instead, courts have relied on the predominant purpose test *to determine whether the Uniform Commercial Code ("UCC") or common law should apply to a given software contract. This test, designed for traditional goods and services that do not share software's complexity or rapid advancement, has perpetuated the uncertainty surrounding software's legal status. This Note proposes that courts adopt the* substantial software test *as an alternative to the predominant purpose test. Under this proposal, the American Law Institute ("ALI")'s* Principles of the Law of Software Contracts *would govern transactions that substantially involve software, and the UCC or common law would govern all other transactions. This new test would provide greater legal clarity with only a minimal shift in jurisprudence. No court has yet adopted a similar test or cited the ALI Principles as authority in a software dispute. The landscape is ripe for change.*

Table of Contents

Introduction 740
I. Resolving Software Disputes Has Become an Epicenter of Uncertainty 742
II. The Predominant Purpose Test Is the Principal Cause of Software's Uncertain Legal Status 745
A. *A Deficit of Appellate Guidance Has Produced Lower Court Uncertainty* 746
B. *Courts Consider Improper Factors in Their Predominant Purpose Analyses* 748
C. *Courts Reason by Analogy in Software Disputes Using Limited and Unsound Precedent* 750
III. The Substantial Software Test Clarifies Software's Legal Status While Exacting Only a Small Shift in Jurisprudence 752

* J.D. Candidate, May 2015, University of Michigan Law School. I am grateful to Professor Bruce W. Frier for his guidance and feedback in developing this Note. Special thanks to the *Michigan Law Review* Notes Office and the team of citecheckers, pageproofers, and production managers for their tireless efforts. I dedicate this Note to my parents, Richard and Alison Gottlieb, and my sister, Jamie, whose love and support have made this journey possible.

A. *The Substantial Software Test and the ALI Principles Provide a Straightforward Framework for Resolving Software Disputes* 754
B. *The Substantial Software Test Is Superior to Alternative Reform Efforts and Overcomes Its Limitations*..... 756
Conclusion 759

Introduction

In 1983, the Soviet Union shot down Korean Air Lines Flight 007, obliterating the aircraft and killing all 269 passengers and crew on board.[1] Flight 007 had entered restricted airspace over Russia "likely because of an incorrect setting on the plane's autopilot" software.[2] In 2009, Air France Flight 447 crashed into the Atlantic Ocean after pilots failed to respond adequately when ice crystals outside caused the plane's autopilot software to disengage.[3] And in 2013, "bad software design" contributed to the runway crash that killed three passengers aboard an Asiana Airlines flight.[4] In an age where software dominates commercial life, it remains unclear what law a court would apply in contract actions arising out of events like these.

There are two primary options. A court would apply Article 2 of the Uniform Commercial Code ("UCC" or "Article 2") if it deemed the autopilot software to be a good, or it would apply common law if it deemed the software to be a service.[5] The difference is not merely semantic. The UCC and common law differ in significant ways on "contract formation and interpretation rules."[6] For instance, a software seller must tender a "perfect" product free of defects under the UCC[7] but must only "substantially" perform under common law.[8] An aggrieved software buyer is consequently more likely to recover under the UCC than under common law when a glitch causes a catastrophic accident.

---

1. Rob Verger, *Newsweek Rewind: When Korean Air Lines Flight 007 Was Shot Down*, Newsweek (July 17, 2014, 5:46 PM), http://www.newsweek.com/newsweek-rewind-when-korean-air-lines-flight-007-was-shot-down-259653.

2. *Id.*

3. Nicola Clark, *Report Cites Cockpit Confusion in Air France Crash*, N.Y. Times, July 6, 2012, at A9, *available at* http://www.nytimes.com/2012/07/06/world/europe/air-france-flight-447-report-cites-confusion-in-cockpit.html.

4. Matthew L. Wald, *Airline Blames Bad Software in San Francisco Crash*, N.Y. Times, Apr. 1, 2014, at A14, *available at* http://nytimes.com/2014/04/01/us/asiana-airlines-says-secondary-cause-of-san-francisco-crash-was-bad-software.html.

5. The UCC applies only to "transactions in goods." *See* U.C.C. § 2-102 (2013).

6. James J. White & Robert S. Summers, Uniform Commercial Code: Revised Article 1 and Amended Article 2—Substance and Process Supplement § 2-2, at 54 (2005).

7. U.C.C. § 2-601.

8. *See, e.g.*, Jacob & Youngs, Inc. v. Kent, 129 N.E. 889 (N.Y. 1921) (holding that a builder is not liable for failing to install a particular brand of piping when the builder used a similar brand of equal quality).

Courts routinely apply the *predominant purpose test*[9] to software contracts to determine if the UCC applies.[10] Under that test, Article 2 governs when the transaction at issue is *predominantly* for goods, while common law applies when the transaction is *predominantly* for services.[11] The U.S. Supreme Court has yet to rule whether software is a good or service, and "there is no national consensus" on the issue.[12] And yet despite its prevalence, the predominant purpose test has failed to assist courts in adjudicating software contract disputes. As a result, software's legal status remains a fundamental yet unanswered question.

This is the first piece of commentary to focus exclusively on the predominant purpose test's limitations in software disputes. It is also the first to present a practical replacement: courts should adopt the *substantial software test*, which would produce tremendous benefits to the legal community while exacting only a small shift in jurisprudence. The substantial software test directs courts away from classifying software contracts as goods or services transactions and asks only whether software is "substantial" in a contract. Courts would then apply the American Law Institute ("ALI")'s *Principles of the Law of Software Contracts* in cases where software is substantial and revert to the UCC or common law for all other contracts. "The Principles are not 'law,' of course, unless a court adopts a provision. Courts can also apply the Principles as a 'gloss' on the common law, UCC Article 2, or other statutes."[13] For courts even to acknowledge the ALI Principles in this context would, in itself, be a giant step forward.

The substantial software test and ALI Principles would bring needed clarity and predictability to software disputes. Almost two million American jobs and $300 billion of the United States' GDP depend on the software

9. Courts vary in using the terms *predominant* and *predominate* in reference to this test. There is no legal distinction between the two, although *Garner's Modern American Usage* describes *predominate* as a "needless variant" of *predominant*. Bryan A. Garner, Garner's Modern American Usage 650 (3d ed. 2009).

10. *See* 1 E. Allan Farnsworth, Farnsworth on Contracts § 1.9, at 44 (3d ed. 2004) ("Courts usually determine whether a transaction is one in goods, services, or land by looking for the 'predominant factor' of the contract."); 1 Howard O. Hunter, Modern Law of Contracts § 9:11, at 519–20 (2014) ("The general test for U.C.C. coverage is to determine whether the primary purpose and main thrust of the contract is the sale of goods or services.").

11. *See supra* note 10. The UCC is not a statute and has no independent legal force, but every state except Louisiana has approved a variant of Article 2. *See* 24 Dian Tooley-Knoblett & David Gruning, Louisiana Civil Law Treatise § 1:4 (2012).

12. *See* Richard Raysman & Peter Brown, *Applicability of the UCC to Software Transactions*, N.Y. L.J., Mar. 8, 2011, at 5.

13. Bob Hillman & Maureen O'Rourke, *American Law Institute Approves the Principles of the Law of Software Contracts*, Concurring Opinions (June 2, 2009), http://www.concurringopinions.com/archives/2009/06/american-law-institute-approves-the-principles-of-the-law-of-software-contracts.html#sthash.skNyh8AP.dpuf.

industry.[14] Too much rests on software's legal status for this issue to remain unsettled.

This Note argues that courts should abandon the predominant purpose test in determining what law governs software contracts and instead utilize the substantial software test. Part I discusses software's uneasy relationship with the UCC and highlights what separates software contracts from other transactions. Part II elaborates on how the predominant purpose test has prompted courts to misapply multifactor balancing tests and haphazardly reason by analogy. Part III proposes the substantial software test and the ALI Principles as an alternative to the predominant purpose test and as an improvement on earlier efforts to clarify software's legal status.

I. Resolving Software Disputes Has Become an Epicenter of Uncertainty

The predominant purpose test is favored for its versatility and simplicity,[15] but the software setting constrains its effectiveness. The test's precise phrasing varies among courts but invariably focuses on whether a contract is predominantly a transaction in goods or in services.[16] The UCC governs a contract that is primarily for goods, even if the contract also includes services, and the common law governs a contract that is primarily for services, even if the contract also includes goods.[17] The test has been so widely adopted in contract law that commentators have evaluated its potential use in other areas of law.[18]

Understanding why the predominant purpose test is inappropriate for software contracts first requires looking at the UCC's scope. The UCC defines goods as "all things (including specially manufactured goods) which

14. *See Software Industry Facts & Figures*, Bus. Software Alliance, http://www.bsa.org/country/public%20policy/~/media/files/policy/security/general/sw_factsfigures.ashx (last visited Oct. 22, 2014).

15. *See, e.g.*, R. Alan Pritchard, *The Predominant Factor Test Under the Uniform Commercial Code*, Tenn. B.J., July 2001, at 23, 26–28 (describing the test's application in a Tennessee appeals court case).

16. *See* Bonebrake v. Cox, 499 F.2d 951, 960 (8th Cir. 1974).

17. *Id.* at 958–60. A Georgia appeals court offered the following iteration: "In a hybrid contract for both goods and services, where the predominant element is the furnishing of services, the Georgia UCC is inapplicable." McCombs v. S. Reg'l Med. Ctr., Inc., 504 S.E.2d 747, 749 (Ga. Ct. App. 1998). The Illinois Supreme Court approached the issue from the opposite angle, holding that the UCC is applicable where "a mixed contract is predominantly for goods and only incidentally for services." Brandt v. Bos. Scientific Corp., 792 N.E.2d 296, 303 (Ill. 2003).

18. *E.g.*, Adam Linkner, *How* Salazar v. Buono *Synthesizes the Supreme Court's Establishment Clause Precedent into a Single Test*, 25 Emory Int'l L. Rev. 57, 59 (2011) (describing the Supreme Court's use of the predominant purpose test in a religious freedom case); Michael S. Kruse, Note, *Missouri's Interfacing of the First Amendment and the Right of Publicity: Is the "Predominant Purpose" Test Really That Desirable?*, 69 Mo. L. Rev. 799, 810, 815–16 (2004) (suggesting that the predominant purpose test may "have the effect of chilling a great deal of artistic expression" if adopted in First Amendment cases).

are movable at the time of identification to the contract for sale."[19] This includes obvious subsets like furniture, toaster ovens, and baseball gloves—standardized products that are easy to conceptualize and evaluate. By contrast, "where the subject matter is intangible, dynamic, and protean," as in the case of software, "proper classification becomes much more problematic."[20] Determining whether the UCC or common law applies to a software lawsuit theoretically should be simple under the binary predominant purpose test, but courts have been unable to use this test uniformly to classify software transactions as involving either goods or services.

Software's amorphous character has plagued courts for decades. One author contends that "[t]he legal definition of software first became an issue in 1969 when IBM . . . announced that it was separating the pricing of its software and services from the pricing of its hardware."[21] The government and courts then had to determine whether "software was tangible personal property and therefore taxable, or intangible intellectual property not subject to taxation."[22]

Software has given rise to a host of other debates in subsequent years. Commentators disagree over whether software is a good or a service;[23] whether, if software is neither a good nor a service, it is instead information;[24] whether software should be subject to federal copyright law rather than state contract law;[25] and whether software transactions involve licenses rather than sales.[26] This Note reframes these questions by focusing instead

---

19. U.C.C. § 2-105(1) (2013).

20. Nancy S. Kim, *Expanding the Scope of the* Principles of the Law of Software Contracts *to Include Digital Content*, 84 Tul. L. Rev. 1595, 1597 (2010).

21. James A. Mogey, *Software as UCC Goods: A Critical Look*, 34 How. L.J. 299, 299–300 (1991).

22. *Id.* at 300.

23. *Compare* Lorin Brennan, *Why Article 2 Cannot Apply to Software Transactions*, 38 Duq. L. Rev. 459 (2000) (arguing that software does not fall within the scope of the UCC), *and* Holly K. Towle, *Enough Already: It Is Time to Acknowledge that UCC Article 2 Does Not Apply to Software and Other Information*, 52 S. Tex. L. Rev. 531 (2011) (same), *with* Sarah Green & Djakhongir Saidov, *Software as Goods*, 2007 J. Bus. L. 161, 161 (arguing that software should be classified as a good under the U.K. Sale of Goods Act 1979 and the U.N. Convention on Contracts for the International Sale of Goods), *and* Bonna Lynn Horovitz, Note, *Computer Software as a Good Under the Uniform Commercial Code: Taking a Byte Out of the Intangibility Myth*, 65 B.U. L. Rev. 129, 162 (1985) ("[S]oftware should be given the status of a good for purposes of applying Article 2 of the UCC.").

24. *See* Brennan, *supra* note 23, at 465–69 (arguing that a sale of software is not a sale of goods but rather is "a license of information").

25. *See* Jean Braucher, *Contracting Out of Article 2 Using a "License" Label: A Strategy that Should Not Work for Software Products*, 40 Loy. L.A. L. Rev. 261, 271–75 (2006) (discussing the question of whether state contract law or federal copyright law—or both—should apply to copy and use restrictions in software license agreements).

26. *Compare* Vernor v. Autodesk, Inc., 621 F.3d 1102, 1111 (9th Cir. 2010) (holding that a software transaction is a license, not a sale of a copy, if the software agreement "(1) specifies that the user is granted a license; (2) significantly restricts the user's ability to transfer the software; and (3) imposes notable use restrictions"), *with* UMG Recordings, Inc. v. Augusto, 628 F.3d 1175, 1177 (9th Cir. 2011) (holding that the distribution of free CDs "effect[ed] a

on the *process* of deciding what law should apply to software contracts. Only when courts have a clear framework for interpreting software contracts will parties know how to design their transactions.

Four issues common to all software contracts shed light on the predominant purpose test's limitations. First, speaking of "software" is like speaking of "food." Two randomly selected examples are bound to be drastically different. Even the definition of software varies greatly from source to source. One possibility—that software is "a series of step-by-step instructions which tell the computer exactly what to do"[27]—is a definition "used in most of the cases and by the commentators."[28] By contrast, *Merriam-Webster* defines software in broader terms, as "the entire set of programs, procedures, and related documentation associated with a system and especially a computer system."[29] These definitional variations are legally significant. Software makers, for example, can patent a program that constitutes a "design or a physical invention" but not a step-by-step "process implemented on a computer."[30]

Second, software varies in scope and power. Consumers who purchase Angry Birds for their tablet device have different expectations of how that product will function than do the purchasers of custom billing software. Developing legal uniformity among the vast swath of products lumped under the umbrella term of "software" has proved exceedingly difficult.[31]

Third, unlike typical goods and services that retain a constant or similar character over time, software undergoes rapid and frequent change. In 1950, Alan Turing envisioned a machine that could simulate human thought, a radical conception at the time.[32] Scholars today ask not only whether computers can think and speak like humans[33] but also whether software can have

---

sale of the discs to the recipients," who were then free to dispose of the CDs without infringing the distributor's copyright).

27. Mogey, *supra* note 21, at 308 (quoting Black's Law Dictionary 633 (5th ed. 1983)) (internal quotation marks omitted).

28. *Id.* (footnote omitted).

29. Webster's Third New International Dictionary of the English Language Unabridged 130a (2002).

30. *See* Adi Robertson, *Supreme Court Rules Software Patents That Cover "Abstract Ideas" Are Invalid*, Verge (June 19, 2014, 11:07 AM), http://www.theverge.com/2014/6/19/5824144/supreme-court-rules-software-patents-that-cover-abstract-ideas-are.

31. Jeffrey B. Ritter, *Software Transactions and Uniformity: Accommodating Codes Under the Code*, 46 Bus. Law. 1825, 1852–53 (1991) (noting the difficulty in harmonizing "different results" when "the facts of two or more [software] cases were nearly comparable").

32. *See* A.M. Turing, *Computing Machinery and Intelligence*, 59 Mind 433, 433 (1950).

33. Apple's iPhone 4s model introduced the Siri app, a software program that can respond to verbal questions on topics ranging from the weather in Phoenix to the nearest supermarket. *See Siri: Your Wish Is Its Command*, Apple, http://www.apple.com/ios/siri/ (last visited Oct. 22, 2014).

moral responsibility.[34] Software has become so powerful that a single malfunction, malicious attack, or instance of human error could irreparably damage global commerce.[35]

Finally, software transactions are typically "mixed" transactions, involving both tangible goods and intangible services like installation, support, and maintenance. Courts have applied Article 2 "even in cases where it is fairly debatable whether there are any goods at all in the transaction," partially because of an inability to differentiate between software goods and services.[36] Courts' difficulty with these mixed transactions has led to a "lack of uniformity" and "an increasing lack of clarity" for the legal community.[37]

These characteristics illustrate why software contracts cannot fit within the confines of traditional contract law. Software products barely resemble one another, they evolve faster than do products in other industries, and they often adjoin other goods and services in hybrid contracts. The conventional predominant purpose test analysis does not account for these qualities. Against this backdrop, it is no surprise that the predominant purpose test is "[bound to] mislead, or eventually lead[ ] to . . . 'pretty awful results'" when applied to software contracts.[38]

II. The Predominant Purpose Test Is the Principal Cause of Software's Uncertain Legal Status

The predominant purpose test's flaws predate software's meteoric rise in the 1990s.[39] Scholarship in the *Harvard Law Review* outlined the test's weaknesses in 1982, well before software became ubiquitous in commercial life.[40]

---

34. Merel Noorman, *Computing and Moral Responsibility*, Stan. Encyclopedia Phil., http://plato.stanford.edu/entries/computing-responsibility/ (last updated Apr. 18, 2014) (rebutting claims that software lacks "causal contribution" to events, cannot "consider the outcomes" of its actions, and has no independent "freedom to act").

35. *See, e.g.*, Retro Report, *The Day the Lights Went Out*, N.Y. Times (Nov. 11, 2013), http://www.nytimes.com/video/us/100000002544427/the-day-the-lights-went-out.html (connecting human failure to activate software with the massive 2003 power blackout in the United States and Canada).

36. 1 William D. Hawkland & Linda J. Rusch, Hawkland's Uniform Commercial Code Series § 2-102:2 (Frederick H. Miller ed., 2014), *available at* Westlaw.

37. Ritter, *supra* note 31, at 1831.

38. Towle, *supra* note 23, at 558 (quoting K.N. Llewellyn, *The First Struggle to Unhorse Sales*, 52 Harv. L. Rev. 873, 873 (1939)).

39. *See* Note, *Disengaging Sales Law from the Sale Construct: A Proposal to Extend the Scope of Article 2 of the UCC*, 96 Harv. L. Rev. 470, 477–78 (1982).

40. *Id.* at 478. Although computers "start[ed] to become a part of daily life for some office workers" in the 1980s, the following decade ushered in an "era of fax/modems, email, the new online world, and dazzling multimedia games and educational software." *A History of Windows*, Windows, http://windows.microsoft.com/en-us/windows/history#T1=era4 (last visited Oct. 22, 2014). The world was introduced to Java, "the most popular computer programming language," in 1995. Francis M. Allegra & Daniel B. Garrie, Plugged In: Guidebook to Software and the Law § 2:2 (2014), *available at* Westlaw. A researcher at the European Organization for Nuclear Research created the precursor of the World Wide Web in 1990. *Id.* § 3:2(B).

The predominant purpose test subjects nontraditional sales, like contracts for both engineering services and construction materials, to "outdated" and "inappropriate" common law; it motivates courts to "distort the facts" so that the UCC will govern a transaction; it induces courts to engage in "unstructured" balancing tests; and it overlooks that a transaction will reflect the UCC's "sale paradigm" to varying degrees as the transaction progresses.[41]

These limitations are serious enough in isolation; software's characteristics only exacerbate their effect.[42] Software is more complex, more abstract, and undergoes more rapid advancement than virtually any other product.[43] "Computer programs are the most intricate, delicately balanced and finely interwoven of all the products of human industry to date."[44] Unlike for boats or bridges, "any architecture, design, or diagram we create for software is essentially inadequate" because it omits fundamental details that render blueprints useless.[45] And although roads have been built "for thousands of years" and have remained mostly constant in design, the first widely used enterprise application framework appeared in 1998 and is now obsolete.[46]

A half century of case law has highlighted how the predominant purpose test is not merely problematic but also unworkable for software disputes. Section II.A examines how a lack of appellate court guidance on software's legal status has caused ongoing judicial uncertainty in using the predominant purpose test. Section II.B then explains why multifactor balancing tests, often used within predominant purpose analyses, have produced significantly worse outcomes in software cases than in other disputes. Finally, Section II.C explores the common law tradition of reasoning by analogy to illustrate its deleterious effects on software law.

A. *A Deficit of Appellate Guidance Has Produced Lower Court Uncertainty*

The predominant purpose test has perpetuated judicial uncertainty in software disputes. In 2002, then–Judge Sotomayor of the Second Circuit remained unsure of whether "U.C.C. Article 2 . . . applies to the licensing of software that is downloadable from the Internet."[47] In *Specht v. Netscape Communications Corp.*, the court stated that it "need not decide today

41. Note, *supra* note 39, at 478; *see* Lincoln Pulp & Paper Co. v. Dravo Corp., 436 F. Supp. 262, 275 (D. Me. 1977).

42. George Stepanek, Software Project Secrets: Why Software Projects Fail 7–21 (2005) (presenting twelve "differences between software development and other common business endeavors").

43. *Id.* at 8–13.

44. James Gleick, *Chasing Bugs in the Electronic Village*, *in* What Just Happened: A Chronicle from the Information Frontier 15, 19 (2002).

45. Stepanek, *supra* note 42, at 11.

46. *Id.* at 13.

47. Specht v. Netscape Commc'ns Corp., 306 F.3d 17, 29 n.13 (2d Cir. 2002).

whether" the UCC applied to downloadable software licenses since, "for present purposes," both approaches would lead to the same outcome.[48] As a result, the case did not instruct lower courts how to rule in software cases when the decision to apply sales law or common law *would* be outcome determinative.

The Second Circuit in *Specht* should have taken a more assertive stance on software's legal status. The court recognized that some courts were indiscriminately choosing whether to apply the UCC to software contracts involving other products.[49] Judge Sotomayor even quoted a district court opinion asserting that "Article 2 technically does not, and certainly will not in the future, govern software licenses, but for the time being, the Court will assume that it does."[50] The Second Circuit's refusal to apply the predominant purpose test in *Specht* deprived lower courts of a signal case containing clear guidance. When the circuit courts forgo deciding these issues—even where doing so would create a circuit split—the district courts are left with little direction and few boundaries.[51]

This judicial uncertainty comes at a significant cost. A business's duty to collect taxes from online sales can hinge on whether those sales involve goods.[52] Court decisions illustrate that a given software product will not uniformly be treated as a good or a service, and thus software sellers may not collect sales taxes when they are required to do so. "[A] four figure tax liability" resulting from this confusion "can be enough to put [a small company] out of business."[53] This example demonstrates a larger concern: "Software transferors and copyholders of all types can perform their various roles confidently and efficiently only after the clarification of applicable rules."[54] Courts should finally acknowledge that the predominant purpose test is unfit for software contracts instead of perpetuating this unworkable regime.

---

48. *Id.*

49. *Id.* ("Some courts have also applied Article 2, occasionally with misgivings, to sales of off-the-shelf software in tangible, packaged formats.").

50. *Id.* (quoting i.Lan Sys. v. Netscout Serv. Level Corp., 183 F. Supp. 2d 328, 332 (D. Mass. 2002)) (internal quotation marks omitted).

51. *See, e.g.*, Phoenix Solutions, Inc. v. Sony Elecs., Inc., 637 F. Supp. 2d 683, 695 (N.D. Cal. 2009) (denying summary judgment on the grounds that the predominant purpose of a software contract was an "unresolved factual dispute[ ]").

52. *See New Tax Rules for Sales of Computers, Software, Digital Goods and Services Takes Effect March 28*, Wash. Tech. Industry Ass'n (Mar. 19, 2013), http://www.washingtontechnology.org/new-tax-rules-for-sales-of-computers-software-digital-goods-and-services-takes-effect-march-28/.

53. *Id.*

54. Principles of the Law of Software Contracts 1, 1 (2010) [hereinafter ALI Principles].

B. *Courts Consider Improper Factors in Their Predominant Purpose Analyses*

Multifactor balancing tests also hamper courts' predominant purpose analyses in software disputes. Courts often utilize such multifactor tests to settle standard-based questions, such as whether a contract is "predominantly" for goods or services. Balancing tests provide courts with a high degree of discretion, which can make the tests "unavoidably vague" and "loosely defined," leading to "inconsistent results."[55]

These shortcomings are particularly true in software cases because courts struggle to conceptualize how software contracts operate, such as when courts weigh how "incidental" or "ancillary" services or goods are to a given software contract.[56] The UCC does not govern a contract that "incidentally involves the supply of goods," nor does it exclude a contract that "incidentally requires the furnishing of services."[57] The more incidental goods are to a contract, however, the more likely it is that services are predominant and vice versa. The inherent vagueness of what can be considered incidental is amplified in the software realm. Courts do not simply disagree about how much is too much; they are inconsistent about what they are measuring and against what benchmark.

Wachter Management Co. v. Dexter & Chaney, Inc. demonstrates that measuring incidental services is a poor balancing factor in software cases.[58] The Kansas Supreme Court in that case determined that the sale of project-management software primarily involved a sale of goods and that the seller's installation, maintenance, and ongoing training were "incidental" to the contract because the services "would not have been necessary" without the purchase of the management software.[59] Following this logic, goods are incidental to a contract when they are unnecessary without the purchase of certain services. *Wachter* therefore suggests that goods and services that are both necessary to a contract are also both incidental.

The fallacy in this definition is obvious when, for instance, a consumer purchases a cable box and television-station package for one price. Surely the cable box is necessary for the consumer to enjoy the television stations, but the television stations are also necessary for the consumer to enjoy the

---

55. David Crump, *Takings by Regulation: How Should Courts Weigh the Balancing Factors?*, 52 Santa Clara L. Rev. 1, 2–4 (2012) (describing the Supreme Court's balancing test for property "takings").

56. *See, e.g.*, Lake & Piepkow Farms v. Purina Mills, Inc., 955 F. Supp. 791, 794 (W.D. Mich. 1997); Kline Iron & Steel Co. v. Gray Commc'ns Consultants, Inc., 715 F. Supp. 135, 139 (D.S.C. 1989); Neibarger v. Universal Coops., Inc., 486 N.W.2d 612, 622 (Mich. 1992).

57. *See* 2 Farnsworth on Contracts, *supra* note 10, § 6.6, at 142–43.

58. 144 P.3d 747 (Kan. 2006).

59. *Wachter*, 144 P.3d at 751. This was the only basis for the court's determination that the services were incidental. *See id.* Services can still constitute a contract's predominant purpose even if the software involved is incidental. *See* Mortg. Plus, Inc. v. DocMagic, Inc., No. 03-2582-GTV-DJW, 2004 WL 2331918, at *3–4 (D. Kan. Aug. 23, 2004).

cable box. Both the cable box and the television stations would be incidental according to *Wachter*, leaving one question: "Incidental to what?"

Some courts entirely fail to explain why goods or services are incidental to a software contract.[60] One court held, without explanation, that services were ancillary to a particular software contract,[61] while another court listed a host of contract principles before summarily "conclud[ing], as a matter of law, that the contract was predominantly for goods and only incidentally for services."[62] This judicial side step is common because, while segregating software into goods and services is challenging, determining their relative weight is nearly impossible.

Courts also give undue weight to the cost or compensation structure of software agreements.[63] One party's cost is typically another party's compensation. They are opposite sides of the same coin. The allocation of costs between goods and services can indicate whether goods or services predominate nonsoftware contracts, and for this reason "courts have placed great weight on the . . . manner in which the contract was billed."[64]

Unlike products in traditional contracts, software often merges the contract's goods and services.[65] The installation of a bowling alley, for example, is wholly separate from the bowling alley itself.[66] It is plausible to assign different cost values to the installation and the bowling alley since they are conceptually distinct. The installation ends where the finished product begins. Parties can argue over the precise allocation of costs, but all stakeholders are operating on common ground.

Software contracts do not involve distinct goods and services. Software is "a codification of the behaviors that the programmers and users want to take place."[67] The good's purpose is to capture and repeat the service. As a result, determining whether a project's costs should be allocated more heavily to the goods or services portion of a software contract will necessarily produce an arbitrary result.[68] Courts' use of software-agreement cost or

60. *See, e.g.*, Surplus.com, Inc. v. Oracle Corp., No. 10-CV-03510, 2010 U.S. Dist. LEXIS 136254, at *15 (N.D. Ill. Dec. 23, 2010); Dealer Mgmt. Sys. v. Design Auto. Grp., Inc., 822 N.E.2d 556, 561 (Ill. App. Ct. 2005).

61. *Surplus.com, Inc.*, 2010 U.S. Dist. LEXIS 136254, at *15.

62. *Dealer Mgmt. Sys.*, 822 N.E.2d at 561.

63. *See, e.g.*, Fab-Tech, Inc. v. E.I. DuPont de Nemours & Co., 311 F. App'x 443, 445 (2d Cir. 2009); Advent Sys. Ltd. v. Unisys Corp., 925 F.2d 670, 676 (3d Cir. 1991); Executone of Columbus, Inc. v. Inter-Tel, Inc., 665 F. Supp. 2d 899, 908 (S.D. Ohio 2009); TK Power, Inc. v. Textron, Inc., 433 F. Supp. 2d 1058, 1062 (N.D. Cal. 2006); Micro-Managers, Inc. v. Gregory, 434 N.W.2d 97, 100 (Wis. Ct. App. 1988).

64. Raysman & Brown, *supra* note 12.

65. *Id.*; *cf.* Green & Saidov, *supra* note 23, at 161 ("[The] unique characteristics [of software] mean that it is not truly analogous to any conventional chattel with which the law is familiar.").

66. Bonebrake v. Cox, 499 F.2d 951, 960 (8th Cir. 1974).

67. *See* Stepanek, *supra* note 42, at 10.

68. *See supra* notes 60–62 and accompanying text.

compensation structures further illustrates how multifactor balancing tests negatively influence the predominant purpose test.

C. *Courts Reason by Analogy in Software Disputes Using Limited and Unsound Precedent*

Reasoning by analogy is a staple of common law jurisprudence but a significant factor in why courts should replace the predominant purpose test in software cases. Courts' ability to analogize to and distinguish from cases is only as effective as the precedent that feeds it. *Advent Systems Ltd. v. Unisys Corp.* exemplifies how one opinion can poison the well of precedent.[69] In that case, the Third Circuit held that an agreement involving an electronic document-management system was primarily a sale of goods.[70] The court recognized that the predominant purpose test "has been criticized" but saw "no reason to depart from that practice here."[71] Scholars have attacked the court's subsequent analysis for misunderstanding federal copyright preemption and conflating a physical copy with a copyright.[72] This misapprehension of copyright law heavily influenced the court's reasoning.

Advent Systems then morphed into a landmark decision for courts using the predominant purpose test to resolve software disputes. Courts across the country cited the decision favorably.[73] In *Triple Point Technology, Inc. v. D.N.L. Risk Management, Inc.*, for example, a district court distinguished "ownership" of a computer program from the right to resale discussed in *Advent Systems*.[74] But the court in *Advent Systems* misunderstood the right to resale, and so the *Triple Point Technology* court used faulty reasoning to reach its holding. Decisions that cite *Triple Point Technology* perpetuate this cycle.

Both software and nonsoftware cases suffer from this snowball effect in which poorly reasoned decisions beget more of the same. It is an unavoidable cost of a common law system. But software cases are particularly prejudiced because the subject matter is so broad that courts often cannot find relevant precedent. The Ninth Circuit held in one case that a software package was predominantly a good notwithstanding ancillary services, such as "training, repair services, and system upgrading."[75] The court did not

69. 925 F.2d 670 (3d Cir. 1991).

70. *Advent Sys. Ltd.*, 925 F.2d at 676.

71. *Id.*

72. *See, e.g.*, Brennan, *supra* note 23, at 549–53.

73. *See* Nat'l Data Payment Sys., Inc. v. Meridian Bank, 212 F.3d 849, 856 (3d Cir. 2000); Micro Data Base Sys., Inc. v. Dharma Sys., Inc., 148 F.3d 649, 654–55 (7th Cir. 1998); Rottner v. AVG Techs. USA, Inc., 943 F. Supp. 2d 222, 230 (D. Mass. 2013); Youngtech, Inc. v. Beijing Book Co., No. A-1788-05T3, 2006 WL 3903976, at *5 (N.J. Super. Ct. App. Div. Dec. 29, 2006); Smart Online, Inc. v. Opensite Techs., Inc., No. 01-CVS-09604, 2003 WL 21555316, at *3 (N.C. Super. Ct. June 17, 2003).

74. Triple Point Tech., Inc. v. D.N.L. Risk Mgmt., Inc., No. CIV.A.99-4888WHW, 2000 WL 1236227, at *7 (D.N.J. Apr. 11, 2000).

75. RRX Indus. v. Lab-Con, Inc., 772 F.2d 543, 546 (9th Cir. 1985).

explain why the services were ancillary to the contract,[76] and one commentator criticized the court for taking "a less rigorous approach" to the predominant purpose test.[77]

Courts that do take a more rigorous approach in software cases do not necessarily fare better. The plaintiffs in a proposed federal class action alleged that antivirus software "consistently reported that a tested PC suffered from multiple problems regardless of its actual health."[78] The court, in applying Delaware state law, examined two previous Delaware decisions that dealt with software disputes: one where the court held that a computer hardware lease was predominantly a good[79] and another where the court considered customized software to be a service.[80] The court applied the predominant purpose test while conceding that the former case was "readily distinguishable" and that the latter bore "no resemblance" to the software at issue.[81]

It is not surprising that precedent often bears little resemblance to new cases; "software" is a vast umbrella term covering navigation systems, billing systems, temperature-control systems, iPhone apps, word-processing programs, and much more. Courts that reason by analogy in software disputes do not have enough relevant, sound, and useful decisions for comparison.

Encouragingly, courts may finally be realizing that reasoning by analogy and the predominant purpose test are unsuitable for software cases. In 2009, the chief justice of the Supreme Court of Indiana noted in *Conwell v. Gray Loon Outdoor Marketing Group, Inc.* that it would be easy to assume "that customized software is a service while pre-made software is a good," but he quickly cautioned that, "when courts try to pour new wine into old legal bottles, we sometimes miss the nuances."[82] *Conwell*'s facts were markedly simpler than those in most software cases, and the court "happily" acknowledged that the case did not "include any of the aspects . . . that have complicated resolution of the U.C.C.'s applicability" elsewhere.[83] Still, the court's skepticism of software precedent was a welcome step forward. Other courts should heed the *Conwell* court's wisdom and avoid reasoning by analogy in software disputes.

---

76. *Id.*

77. Lawrence B. Levy & Suzanne Y. Bell, *Software Product Liability: Understanding and Minimizing the Risks*, 5 HIGH TECH. L.J. 1, 5 (1990); *see also* Brennan, *supra* note 23, at 569 (noting the court's failure to explain why the sales aspect predominated the contract).

78. *Rottner*, 943 F. Supp. 2d at 225.

79. *Id.* at 229–30 (citing Neilson Bus. Equip. Ctr., Inc. v. Monteleone, 524 A.2d 1172 (Del. 1987)).

80. *Id.* (citing Wharton Mgmt. Grp. v. Sigma Consultants, Inc., 1990 WL 18360 (Del. Super. Ct. Jan. 29, 1990), *aff'd*, 582 A.2d 936 (Del. 1990) (unpublished table decision)).

81. *Id.* at 230.

82. 906 N.E.2d 805, 812 (Ind. 2009).

83. *Conwell*, 906 N.E.2d at 811.

III. The Substantial Software Test Clarifies Software's Legal Status While Exacting Only a Small Shift in Jurisprudence

The predominant purpose test has failed to provide certainty, cohesion, or an underlying logic in software law. Courts should therefore abandon this test in software disputes and adopt a practical alternative: the *substantial software test*. The substantial software test answers procedurally *when* certain law should apply and substantively *what* law should apply. Under this test, courts would apply the ALI's *Principles of the Law of Software Contracts* when software constitutes a substantial portion of the contract. Courts would revert to applying common law or the UCC in all other cases.

The substantial software test's design is mindful of earlier unsuccessful attempts to improve software law. By the middle of the twentieth century, courts and practitioners encountered "chaos" in "trying to determine which contract law" applied in a particular dispute.[84] A campaign began to revise Article 2 by separating it into three parts with a common core of provisions, known as the "hub-and-spoke" approach, but the effort failed to garner popular support.[85] No state adopted this "revised Article 2."[86]

The Uniform Computer Information Transactions Act ("UCITA") soon emerged as an alternative proposal that focused exclusively on "information transactions," including software contracts.[87] But UCITA was heavily polarizing among the software community.[88] Opponents denounced it as "an extreme measure being advanced by the software industry"[89] that contravened "traditional copyright law"[90] and altered UCC warranties to favor software producers.[91] Thirty-three state attorneys general went on record as opposing UCITA, expressing fears that "consumers under UCITA [would] lose many rights that generally accompany the sales of goods," such as the

84. Towle, *supra* note 23, at 535; *see also* text accompanying note 21.

85. Bruce W. Frier & James J. White, The Modern Law of Contracts 205–06 (3d ed. 2012) ("[A]dvocates for the revision were weak and unenthusiastic, while the opponents were many and more vigorous . . . .").

86. Towle, *supra* note 23, at 533 n.2.

87. *See* Unif. Computer Info. Transactions Act (2002), *available at* http://www.uniformlaws.org/shared/docs/computer_information_transactions/ucita_final_02.pdf.

88. White & Summers, *supra* note 6, at 53 ("Certain groups, particularly consumer advocates, librarians and some licensees regard UCITA as akin to the Communist Manifesto, while others, particularly licensors, regard it as more sacred than the Old Testament.").

89. David A. Szwak, *Uniform Computer Information Transactions Act [U.C.I.T.A.]: The Consumer's Perspective*, 63 La. L. Rev. 27, 51 (2002).

90. Deborah Tussey, *UCITA, Copyright, and Capture*, 21 Cardozo Arts & Ent. L.J. 319, 332 (2003).

91. Braucher, *supra* note 25, at 271; *see also* Ajay Ayyappan, Note, *UCITA: Uniformity at the Price of Fairness?*, 69 Fordham L. Rev. 2471, 2505–10 (2001) (discussing concerns of UCITA's opponents that the statute would weaken consumer warranty protections).

ability to use a product "where and how [consumers] wish."[92] The state legislatures in Iowa, North Carolina, West Virginia, and Vermont even approved "bomb-shelter" statutes that prohibited courts from applying UCITA in contract disputes.[93] Only two states, Maryland and Virginia, ultimately approved UCITA legislation.[94]

In the wake of revised Article 2 and UCITA, the ALI attempted to end the long-standing debate over software's legal status when it approved the ALI Principles in 2009.[95] The ALI had successfully "employed the 'Principles' approach before in projects" involving corporate governance and family dissolution.[96]

The ALI Principles are essentially a nonbinding user's manual on how best to apply software law. The Principles "account[ ] for the case law and recommend[ ] best practices, without unduly hindering the law's adaptability to future developments."[97] Unlike a Restatement project, the Principles outline "the 'black letter' rather broadly, with substantial elaboration in Comments."[98] They focus less on regulating the transfer of software than on encouraging ex ante disclosure of facts, terms, and postcontract intentions.[99] They are not designed for legislative adoption but instead to "give guidance to lawyers, persons in the software business or who rely on software, and eventually common-law judges and legislators."[100] The Principles provide a software-specific approach to contract disputes and a way forward from the archaic goods–services debate.

This Part promotes the substantial software test and the ALI Principles as a permanent replacement for the predominant purpose test and sales or common law. Section III.A assesses the ALI Principles's substantive strength and suggests how courts can capitalize on this strength with the substantial software test. Section III.B rebuts salient counterarguments to adopting the substantial software test.

---

92. Letter from Nat'l Ass'n of Attorneys Gen. to Carlyle C. Ring, Ober, Kaler, Grimes & Shriver (Nov. 13, 2001), *available at* http://www.ucita.com/pdf/Nov132001_Letter_from_AGs_to_Carlyle_Ring.pdf.

93. Nim Razook, *The Politics and Promise of UCITA*, 36 Creighton L. Rev. 643, 644 n.4 (2003); *NCCUSL's Withdrawal of Active Support from UCITA*, Americans for Fair Electronic Com. Transactions (Aug. 4, 2003), http://affect.ucita.com/pdf/AFFECT_8-04-03_PressRelease.pdf.

94. *See* Md. Code Ann., Com. Law §§ 22-101 to -816 (LexisNexis 2013); Va. Code Ann. §§ 59.1-501.1 to -509.2 (2014).

95. Robert A. Hillman & Maureen A. O'Rourke, Principles of the Law of Software Contracts*: Some Highlights*, 84 Tul. L. Rev. 1519, 1519 (2010).

96. ALI Principles, *supra* note 54, at 2.

97. *Id.*

98. *Id.*

99. Robert A. Hillman & Maureen O'Rourke, *Defending Disclosure in Software Licensing*, 78 U. Chi. L. Rev. 95, 95 (2011); *see also* Florencia Marotta-Wurgler, *Will Increased Disclosure Help?: Evaluating the Recommendations of the ALI's "Principles of the Law of Software Contracts"*, 78 U. Chi. L. Rev. 165, 167 (2011) (praising the Principles's preference for disclosure over regulation).

100. ALI Principles, *supra* note 54, at IX–X.

A. *The Substantial Software Test and the ALI Principles Provide a Straightforward Framework for Resolving Software Disputes*

The substantial software test includes both a legal test and a body of law. The legal test's primary strength is its simplicity. Courts need not grapple with whether goods or services predominate a software contract or whether the transfer at issue is a license or a sale.[101] Rather, courts must decide only whether software is a *substantial* component of a given contract. The difficulty inherent in deciding what qualifies as "substantial" is far less complex than the difficulty involved in separating software contracts into those for goods and those for services. And the substantial software test's less demanding threshold—which asks if software is substantial rather than predominant—would allow the ALI Principles to govern as a quasidefault regime. Contract parties could reasonably expect that the Principles would apply in any future litigation, and the parties could plan accordingly.

Replacing the predominant purpose test with the substantial software test would, without further changes, increase legal clarity and predictability. The ALI Principles complement the substantial software test and fill a void left by general contract law.[102] They memorialize existing law[103] but perceive the market as too weak to ensure that consumers receive equitable licensing terms.[104] The Principles offer the benefits of specialized over general contract law,[105] and they deal with perennial software issues like contract formation, standard form terms, and the remote disabling of licensed software.[106] Adopting the Principles would also bring American jurisprudence in line with that of the common law community. The United Kingdom,[107] India,[108]

---

101. Hillman & O'Rourke, *supra* note 95, at 1522.

102. *See* Frier & White, *supra* note 85, at 206 (asserting that the "absence of any . . . reliable source of comprehensive body of doctrine" will likely push courts to adopt the ALI Principles as law).

103. *Cf.* Katharine T. Bartlett, *U.S. Custody Law and Trends in the Context of the ALI Principles of the Law of Family Dissolution*, 10 Va. J. Soc. Pol'y & L. 5, 6 (2002) ("Unlike traditional ALI Restatements, ALI Principles strive to find 'best practices' without necessarily being constrained by existing law.").

104. *See* Marotta-Wurgler, *supra* note 99, at 166.

105. *See generally* Robert A. Hillman, *Contract Law in Context: The Case of Software Contracts*, 45 Wake Forest L. Rev. 669 (2010).

106. Juliet M. Moringiello & William L. Reynolds, *What's Software Got To Do with It? The ALI* Principles of the Law of Software Contracts, 84 Tul. L. Rev. 1541, 1549–52 (2010).

107. The United Kingdom's Intellectual Property Office firmly classifies "computer software" as a good, but it classifies the design, development, installation, and repair of software as a service. *Trade Mark Classification List of Goods and Services*, Intell. Prop. Off. (May 2, 2014), https://www.gov.uk/government/publications/how-to-classify-trade-marks/trade-mark-classification-list-of-goods-and-services (class 9 and class 42).

108. The Madras High Court found that "software [is] undoubtedly goods, as it [is] an article of value having regard to its utility, capability of being bought and sold, and capability of transmission, delivery, storage and possession." S. Madhavan, *Is Supply of Software a Sale or a Service?*, Bus. Standard (Sept. 6, 2010), http://www.business-standard.com/article/economy-policy/is-supply-of-software-a-sale-or-a-service-110090600038_1.html.

and Australia[109] have all taken concrete steps to remove the ambiguity in software's legal status.

Finally, to the extent that the ALI Principles "would be helpful to both the parties and courts in determining outcomes," adopting them would not require a significant shift in jurisprudence.[110] The Principles address topics that neither the common law nor the UCC discusses,[111] and they do not completely overhaul the judicial landscape or render null decades of case law.[112] In short, they offer judges a sophisticated yet accessible guide for resolving software cases.

Despite their suitability, the ALI Principles have thus far been a nonfactor in the courts.[113] In 2009, the Indiana Supreme Court cited the Principles in dicta, merely noting their existence,[114] but no court has used the Principles as authority in any published judicial decision.[115] The most likely explanation for this is that courts, attorneys, and many academics are simply unfamiliar with them. Working drafts of the Principles, unlike drafts for revised Article 2 or UCITA, were not readily accessible online.[116] This "probably shielded the Principles from public scrutiny comparable to that devoted to [revised] Article 2[ ] and UCITA."[117] CNN and the *New York Times* gave UCITA "prominent coverage" during its development, while only "specialty technology journals and Web sites" covered the adoption of the ALI Principles.[118] That they are relatively unfamiliar should not dissuade courts from adopting them, however. The substantial software test and ALI

109. *See, e.g.*, *Competition and Consumer Act 2010* (Cth) sch 2 (Austl.), *available at* http://www.comlaw.gov.au/Details/C2014C00486/Html/Volume_3#_Toc394409742 (defining software as "goods").

110. Maureen A. O'Rourke, *The ALI's Principles of Software Contracting: Some Comments & Clarifications*, 12 J. High Tech. L. 159, 166 (2011).

111. *Id.* at 166–67.

112. Nonetheless, the Principles have not been immune to criticism. One commentator lauded the Principles as "an impressive accomplishment" but was dismayed that they excluded digital content. Kim, *supra* note 20, at 1596–97. Two members of the ALI Principles Members Consultative Group criticized the project for failing to differentiate between licenses and sales and for its lack of guidance on when federal intellectual property law should preempt state contract law. Moringiello & Reynolds, *supra* note 106, at 1541 n.†, 1543–47. Another commentator determined that one of the Principles's provisions, intended to increase consumer contract readership, would yield only a single additional reader for every 278 shoppers. Marotta-Wurgler, *supra* note 99, at 167–68, 183–84.

113. Florencia Marotta-Wurgler & Robert Taylor, *Set in Stone? Change and Innovation in Consumer Standard-Form Contracts*, 88 N.Y.U. L. Rev. 240, 270 n.77 (2013) (observing that "courts do not appear to have relied on [the ALI Principles] yet").

114. Conwell v. Gray Loon Outdoor Mktg. Grp., Inc., 906 N.E.2d 805, 811 & n.7 (Ind. 2009).

115. *See* Principles of the Law of Software Contracts, General Case Citations (West, Westlaw through June 2014).

116. Hannibal Travis, *The* Principles of the Law of Software Contracts*: At Odds with Copyright, Consumer, and European Law?*, 84 Tul. L. Rev. 1557, 1565 (2010).

117. *Id.*

118. *Id.* at 1565 n.40.

Principles together can end the long-standing ambiguity surrounding software's legal status.

B. *The Substantial Software Test Is Superior to Alternative Reform Efforts and Overcomes Its Limitations*

The substantial software test prevails not only over the predominant purpose test but also over alternative efforts to correct software law. Aside from implementing the substantial software test, courts could adopt a hard-line rule. They could also replace the predominant purpose test with an existing, more familiar test. Or they could simply defer to state legislatures and continue to apply the predominant purpose test unless they receive a statutory directive indicating otherwise. All three alternatives are ultimately inadequate when compared with the substantial software test, however.

Courts should resist adopting a per-se rule that either categorically applies the UCC or completely excludes it from consideration. While such a rule would be straightforward and would eliminate any confusion over whether the UCC or common law applies, both variations contain flaws that would render them unworkable in the software context.

Those favoring a narrower construction of Article 2 would bar the UCC from ever applying to software.[119] But courts that invariably apply common law may do so even in instances where the software involved is insignificant.[120] For example, the sale of refrigerators containing embedded computer chips is clearly a transaction in goods and should be governed by the UCC, but a per-se rule applying common law to all software-related transactions would mandate a contrary result.

Other advocates of a per-se rule, by contrast, would apply the UCC in *all* software disputes.[121] But this approach would transform the Code into "the successor of the evil from the past that [UCC drafter Karl Llewellyn] sought to avoid."[122] The UCC was written to govern traditional "wares moving to and through a merchants' market."[123] Llewellyn could not conceive of, and the UCC cannot accommodate, the intricacies of software contracts. The substantial software test is superior to a per-se rule that categorically applies the UCC because it still requires that a contract be *substantially* composed of software.

The substantial software test is also superior to extant legal tests like the *gravamen of the action test* and the *bifurcated transaction test*. The gravamen of the action test focuses on the nature of the *lawsuit* in determining

119. *See* Towle, *supra* note 23, at 532.

120. *See supra* Section II.C.

121. *E.g.*, Braucher, *supra* note 25, at 278. Under this view, doing away with the UCC for all software disputes would sacrifice a "flexible approach to contract formation" and administrable rules on everything from warranties to remedies for nonperformance. *See id.* at 276–77.

122. Raymond T. Nimmer, *Images and Contract Law—What Law Applies to Transactions in Information*, 36 Hous. L. Rev. 1, 18–19 (1999).

123. *See* K.N. Llewellyn, *The First Struggle to Unhorse Sales*, 52 Harv. L. Rev. 873, 883 (1939) (discussing early factors motivating the desire to create a uniform body of sales law).

whether Article 2 should govern the contract dispute.[124] Under the gravamen test, there is no need to determine whether the entire contract focuses predominantly on goods or services. Courts could apply the law "most relevant" to the cause of action.[125]

The substantial software test remains a better option. Bright-line divisions between goods and services lead to "insurmountable problems of proof in segregating assets."[126] The predominant purpose test has failed in large measure precisely because it requires courts to segregate assets. Moreover, the gravamen test, for its part, "has not won widespread acceptance in the courts,"[127] and some states have "explicitly rejected" it.[128] By contrast, the substantial software test avoids the goods–services debate altogether, and it does not have the tarnished reputation of the gravamen test.

The Tenth Circuit in 1967 presented another option, the bifurcated transaction test, but most courts have since rejected it.[129] Unlike the gravamen of the action and predominant purpose tests, which have an all-or-nothing effect, the bifurcation test directs courts to apply the UCC to the goods portion of a transaction and the common law to the services portion.[130] This test therefore enables courts to avoid deciding whether a contract is primarily for goods or for services. And yet most courts have "rejected this approach as unworkable,"[131] because it entails applying

124. Nathalie Martin, Software Transactions *and U.C.C. Article 2*, Emerging Issues, Apr. 27, 2008, *available at* 2008 Emerging Issues 102 (LexisNexis).

125. Towle, *supra* note 23, at 555 n.70; *see also In re* Fort Totten Metrorail Cases Arising Out of the Events of June 22, 2009, 793 F. Supp. 2d 133 (D.D.C. 2011). In 2009, two Washington Metropolitan Area Transit Authority ("WMATA") subway trains collided, leaving nine dead and dozens injured in the worst WMATA accident in D.C. history. The decedent passengers' estates sued WMATA and Alston Signing for, among other things, breach of UCC implied warranties. *See In re Fort Totten*, 793 F. Supp. 2d at 137–38. The plaintiffs alleged that one of the subway's train-detection systems was defective. *Id.* The court concluded that Alston's sale to WMATA of "train traffic control equipment, software and support services" made the company a merchant under the UCC. *Id.* at 151. That the court looked to the specific transfer of software relating to the subway collision speaks to the gravamen test's potential. *See id.*

126. Hudson v. Town & Country True Value Hardware, Inc., 666 S.W.2d 51, 54 (Tenn. 1984).

127. 1 Hawkland's Uniform Commercial Code Series, *supra* note 36, § 2-102:2.

128. Martin, *supra* note 124 (citing appellate courts in Illinois and Indiana); *cf.* George E. Henderson, *A New Chapter 2 for Texas: Well-Suited or Ill-Fitting*, 41 Tex. Tech L. Rev. 235, 280–81 (2009) (suggesting that Texas courts are unlikely to use *any* "goods-services test" to determine whether the UCC applies to software disputes).

129. Martin, *supra* note 124 (citing Foster v. Colo. Radio Corp., 381 F.2d 222 (10th Cir. 1967)).

130. *Id.*

131. *Id.* (citing Rajala v. Allied Corp., 66 B.R. 582 (D. Kan. 1986)); *see also* Consol. Edison Co. of N.Y. v. Westinghouse Elec. Corp., 567 F. Supp. 358, 362 (S.D.N.Y. 1983) ("The New York courts appear to have rejected an approach of applying sales law to the sales aspect of a transaction which combines both sales and service features, requiring instead that the applicable law be determined by looking to the essential nature of the underlying contract.").

"different laws to the same contract."[132] Insisting that courts apply the Statute of Frauds only to the goods portion of a contract would make performance impossible "within the intention of the parties."[133] The bifurcation test also fails to address how courts should treat the software aspect of a mixed contract. The substantial software test, by contrast, involves one body of law that is tailored for software contracts.

The gravamen and bifurcation tests are valuable insofar as they achieve better results in limited circumstances. Courts can effectively apply the gravamen test when a lawsuit's primary focus is clearly goods or services and the bifurcation test when a contract is divisible into goods and services portions. These scenarios are exceedingly rare, however, and courts would still be left to apply ill-fitting UCC provisions or common law. Courts will earn greater long-term dividends by utilizing the substantial software test and the ALI Principles.

Finally, the substantial software test is also preferable to legislative initiatives like UCITA. Courts may be tempted to defer to state legislatures for guidance on software law, but UCITA's demise illustrates how unlikely it is that reform will come from the statehouse. Legislative initiatives allow public opinion and special interest groups to band together in opposition. Neither the ALI Principles nor the substantial software test requires legislative approval, except in Maryland and Virginia, where UCITA governs. Moreover, because the ALI Principles would remain constant, the substantial software test is also more likely than separate state statutes to produce a unified body of case law.

Of course, the substantial software test's relative superiority does not mean it is flawless. Determining whether software is substantial in a contract is subject to judicial discretion and balancing. In making this determination, then, courts may misuse or misapply the ALI Principles, thereby creating another incoherent body of doctrine. And for all its problems in the software setting, the predominant purpose test has worked well in other areas of contract law and could possibly be tweaked to accommodate software's uniqueness.

None of these concerns should ultimately carry the day. Determining whether software is a substantial component of a contract is less complex than segregating contracts into those for goods and those for services. A court needs less technological expertise to decide whether customizable billing software is substantial in a contract than to decide whether that software is predominantly a good or a service. In this sense, the substantial software test asks a more basic and accessible question.

Concerns that courts will misinterpret the ALI Principles are understandable but remain unpersuasive. If necessary, the ALI can always modify the Principles, placing more emphasis on one topic and less on another. In the worst-case scenario, if the Principles are found unworkable, courts will

132. Martin, *supra* note 124.

133. De Filippo v. Ford Motor Co., 516 F.2d 1313, 1323 (3d Cir. 1975).

stop applying them, and the Principles will become a short-lived and unsuccessful experiment. Should they fulfill their stated mission, though, the Principles will define the contours of an entire body of doctrine and immeasurably improve legal clarity.

Lastly, courts and practitioners may be wary of entirely abandoning the predominant purpose test and instead search for a less wholesale change than the substantial software test. But such a search would rest on a mistaken premise. The substantial software test is not a revolutionary overhaul of substantive law. It simply arranges the fragmented pieces of software jurisprudence and adds a modern touch to interpretation. This reality should assuage concerns that adopting the test would constitute a drastic change.

Conclusion

While the predominant purpose test has needlessly perpetuated software's uncertain legal status, the substantial software test can correct decades of judicial confusion over what law should apply in software disputes. Courts that adopt this test would no longer be forced to wrestle with determining the primary purpose of a contract, and they would benefit from the substantive provisions of the ALI *Principles of the Law of Software Contracts*. This Note has argued that courts should no longer hesitate to apply the ALI Principles as authority in software disputes. Instead, they should embrace the substantial software test and the ALI Principles as an important and long overdue step in clarifying software jurisprudence.