Bloomberg Law

Professional Perspective

Understanding Software Developer Liability and Mitigating Legal Risk

Richard Lee, Civis Analytics, and
Justin Steffen and Reena Bajowala, Ice Miller

Understanding Software Developer Liability and Mitigating Legal Risk

Contributed by Richard Lee, Civis Analytics, and Justin Steffen and Reena Bajowala, Ice Miller

Software developers and B2B software companies have largely managed to evade the enormous volume of lawsuits that imperil many consumer-facing businesses due to unique legal defenses and a set of challenging practical circumstances that often make filing a lawsuit not viable. These impediments have led some to suggest the existence of a "tech veil."

Changes in the stream of commerce and statements and actions of regulators, however, portend change. Today, algorithms and artificial intelligence are ubiquitous. Companies are creating code that is packaged, sold, and consumed by businesses and, ultimately, those businesses' customers. Tech philosopher Tom Chatfield implored us to "[f]orget artificial intelligence—in the brave new world of big data, it's artificial idiocy we should be looking out for." Developers and software companies need to understand the legal landscape and, armed with this knowledge, take discrete steps to guard against the prospect of increasing legal liability.

Background

Nothing per se bars plaintiffs from suing software companies or developers. Indeed, developers and other IT professionals can be and are sued under breach of contract and traditional tort theories. See, e.g., *Affinity Gaming v. Trustwave Holdings*, No. 2:15-cv-024 (D. Nev.). A number of legal doctrines and practical issues, however, have combined to shield developers from the expanse of lawsuits that they might otherwise face. This is what Blakemore Fallon attorney Jenny Leung cleverly termed the "tech veil."

First, the economic loss doctrine and the traditional elements required of product liability claims provide developers some cover. Under the economic loss doctrine, for example, a party that sustains only economic injuries is limited to contract claims and cannot recover for a tort, such as negligence. *Chevron USA, Inc. v. Aker Maritime, Inc.*, 604 F.3d 888, 900 (5th Cir. 2010). While tort law might allow a claim to proceed if the developer owed a special duty towards the plaintiff, standards of professional negligence in the IT world are a roadblock.

Courts that have considered the issue have typically held that software developers or IT consultants are not "professionals" who owe an enhanced duty of care under the law. See *Avazpour Networking Servs. v. Falconstor Software, Inc.* (S.D.N.Y.) and *Ferris & Salter P.C. v. Thomson Reuters Corp.* (D. Minn.) Likewise, products liability claims typically require some form of physical harm. *Mink v. Univ. of Chicago*, 460 F. Supp. 713, 719 (N.D. Ill. 1978). With some exceptions, many algorithms and software products will not cause bodily injury or property damage.

The difficulties inherent in proving causation provide additional protection to developers, and hurdles to plaintiffs. To establish liability under a variety of legal theories, plaintiffs need to prove proximate causation. In other words, plaintiffs have to establish that a purported technology defect or issue caused the damage, absent any intervening causal factors. With layers of parties standing between the developer and the consumer, this obstacle is a difficult one to overcome.

Complicating the issue of causation further, a number of developers are creating products using different forms of artificial intelligence. While the term "artificial intelligence" is more a marketing construct than one-size-fits-all definition, neural networks, support vector machines, or other deep-learning algorithms present a black box problem. It becomes increasingly difficult to determine precisely how these products reached a decision, thus making establishing causation an even more difficult task.

As a result of the above, plaintiffs—such as downstream consumers who have no contractual relationship with developers—may not succeed on tort or products liability claims against developers.

Second, practical considerations have contributed to the "tech veil." As noted above, developers are increasingly providing specialized products to consumer-facing businesses that incorporate the developer's work into a product that is then sold to consumers or other clients. When the developer and the consumer- or client-facing business are not the same entity,

however, the consumer or end client does not have a contract with the developer and therefore no right to sue under the same. When the zipper on your Prada bag breaks, you take the bag back to Prada, not to YKK. Most times, moreover, the deep pockets that plaintiffs target are the companies selling the product, not the developers themselves.

In addition, developers of both open-source licenses and proprietary software include limitations of liability in their licenses and contracts that bar consequential damages, insulating themselves from certain business-to-business legal exposure.

Regulators, likewise, while cognizant of vendor relationships, tend to focus on the regulated entities themselves and those entities bear the brunt of regulatory oversight. Bank of America, for example, has to mind the Office of the Comptroller of the Currency, the Federal Trade Commission, the Consumer Fraud Protection Bureau, and other state and federal regulators. The cybersecurity company that sells software to Bank of America is simply not subject to the same level of scrutiny.

But like all things technology, time does not stand still. Changes in the developer ecosystem and recent statements and actions from regulators suggest increased legal risk for developers. The stream of commerce has changed, for the better, for many developers. First, developers, IT consultants, and other related businesses have not only created a separate set of companies distinct from the consumer-facing businesses that they serve, but these businesses have also become quite successful. When trouble arises, these now well-funded businesses will invariably be involved in more litigation. They are no longer "judgment-proof."

Pronouncements and actions from regulators also signal that the developers themselves will be subject to more scrutiny going forward. In an Oct. 2018 speech, for instance, Commodity Futures Trading Commissioner Brian Quintenz raised the question of whether smart contract developers could be liable for prediction contracts that fail to comply with CFTC rules. Similarly, Treasury, the FTC, and myriad other government agencies have prepared reports, conducted hearings, and/or examined the impact of and risks posed by new technologies.

Potential Areas of Developer Liability

In this new stream of commerce, developers must recognize legal exposure in at least four distinct contexts: business-to-business; downstream, consumer or client liability; regulatory risk; and other potential plaintiffs.

The business-to-business legal risks have not changed much. Developers are still subject to breach of contract and tort claims from their contractual counterparties. Moreover, harkening the debate in the blockchain space as to whether coders are fiduciaries, there may be attempts to impose additional duties (fiduciary, professional, or other) on developers

Notwithstanding some of the hurdles discussed above, consumers can and will assert tort and products claims against developers when algorithms, software, and smart contracts do not perform as expected. Consumers, for example, denied a loan by an online lender using a developer's faulty proprietary credit algorithm could claim to have suffered emotional distress, thereby satisfying any physical harm requirement. Regardless of viability of such a claim, if the business that developed the algorithm is a successful company, they will likely be named as a defendant.

As evidenced by Quintenz's comments, moreover, regulators may subject developers to more demanding scrutiny going forward. Banking regulators like the OCC, for example, are familiar with evaluating banks' vendors and key third-party relationships. Likewise, the broad enforcement authority over unfair and deceptive practices, as granted to the FTC and the Consumer Financial Protection Bureau, could bring developers within their orbit. Abroad, moreover, regulators have issued extensive guidance governing standards for compliance with Article 22 of the General Data Protection Regulation—the provision concerning automated decision-making.

Developers may also have to appreciate risks posed by other plaintiffs. State attorneys general, for instance, may have authority to prevent unfair and deceptive trade practices. Other businesses that contract with a developer's counterparty, moreover, may be impacted by defective algorithms.

Eliminate, Transfer, Mitigate

While the risks are real, developers can take a number of steps to help manage these risks. For purposes of illustrating these steps, consider the following hypotheticals:

- A tech startup creates a proprietary algorithm that evaluates hundreds of data points to make lending decisions. The algorithm "learns," evolving with every credit decision and with more training data. The algorithm is licensed to an online lending company that then makes loans using the algorithm to consumers. A bias in the algorithm—potentially because either the training data was biased, the developer did not properly validate or did not properly address errors when validating the algorithm, or because the developer's subjective definition of "fairness" was not enough to cause the developer to identify errors caused by certain discriminatory algorithmic biases that produced these errors—causes the company to deny credit to customers based on discriminatory factors and to customers who are actually eligible for the loan programs.

- A tech company creates a tool that evaluates patients' medical histories, identifies potential health risks, and suggests treatments. The company sells the tool to hospitals whose doctors use the tool to help treat patients. A defect in the code for the diagnostic tool causes it to misdiagnose a number of patients, subjecting them to unnecessary treatments.

First, the developer in hypotheticals 1 and 2 could have taken additional steps to eliminate the risk. Many pundits speak of the need to test and validate algorithms. Regardless of how the work is performed, significant testing is a highly advisable precaution. In addition, the lack of action may eliminate or mitigate certain risks. If, for example, class action laws are particularly onerous in one jurisdiction, the developer may provide its product to the company on the condition that it is not used for residents of jurisdictions.

Similarly, to avoid unwanted regulatory and legal liability, crypto entrepreneurs frequently utilize geo-fences. In any event, developers should be cognizant of the industries where their products will be deployed and appreciate the risks posed by the regulators responsible for those industries.

Second, developers can seek to transfer their risk. Insurance is a common tool used to mitigate risks. Developers, however, should be careful to evaluate their policies (E&O, CGL, etc.) to determine the scope of their coverage. Not all policies will protect the developer in every instance.

Third, developers can seek to mitigate the risk. Often, careful contract drafting is the most effective means to ensuring this step. A limitation of liability provision or indemnity may absolve the developer of unwanted or unexpected liability.

Developers, however, should be cognizant that some of the aforementioned steps may be inconsistent with each other. For example, an overly strong disclaimer by the developer in hypothetical 1 could be both to the developer's benefit and detriment. A disclaimer that the lending algorithm could result in discriminatory lending decisions or could result in unintended denials of credit could protect the developer against a breach of contract or other claim brought by the company. That same disclaimer could also be cited as evidence by a consumer that the developer foresaw those issues could occur.

Regardless of their approach, developers need to appreciate the risks presented by their products and take affirmative steps to manage those risks. Developers can no longer rely on a non-existing or eroding tech veil. In some ways, they are victims of their own success. Success and potential causes of action will yield lawsuits and developers need to be prepared.