NEAL PARIKH

GINNY FAHS

BRANDIE NONNECKE

To learn more about this project, please visit aspentechpolicyhub.org.

THE ASPEN INSTITUTE

Photo by Photo by Etienne Boulanger on Unsplash

Test-Driven Development for Technology Policy

Adapting computer science methods for building robust and reliable software to policymaking

EXECUTIVE SUMMARY

Technology policy frequently does not work as intended. Often, policies in this space do not fully address the issues they are concerned with or have negative unintended consequences. Developing software was also once as ad-hoc and error-prone until robust methods for testing and writing software were developed. This project recommends applying one such framework, test-driven development (TDD), to policymaking to help develop more robust policies that are more likely to accurately address their issue of concern. The method does not require the use of engineering tools or software; it is simply a methodology that focuses on brainstorming concrete situations in which to 'test' possible policy solutions.

THE PROBLEM

Currently, practices used to create technology policy can be ineffective. A top-down policymaking approach addressing a suite of problems can fail to consider or specify an intended policy's role in specific cases, especially in edge cases or worst-case or adversarial scenarios. This approach also generally does not actively consider cases where the policy should not apply, which can result in the policy being overly broad.

> *Our governments must rethink how they serve the public using the tech-enabled feedback and iterative improvement mechanisms that drive some of the most successful consumer and enterprise services today.*
>
> Jennifer Pahlka

THE SOLUTION

This project proposes adapting a method called test-driven development (TDD) for crafting technology policy. TDD (counterintuitively) advises writing tests first, only then writing the software so it satisfies the requirements laid out in the test. Each time the software is changed, the suite of tests is rerun to ensure the software still complies.

In the policymaking case, policymakers would begin by brainstorming "tests," which are descriptions of what is or is not supposed to happen in concrete situations as a result of the policy. Only then is actual policy language drafted, and this is checked against the tests as the policy language is revised. This systematic, iterative process helps policymakers avoid mistakes when drafting policy. It also has many other benefits, such as it being easier to get technologists or domain experts to contribute tests rather than work directly on abstract or legal policy language.