6.5.2 Build an app and hope it dominates dissapears

PROMPT:

Consider the following world:
1. Main digital development happens through spec driven AI assisted development
2. All applications are open source
3 These open source projects are maintained by humans using spec driven development and afterwords human discussion regarding implementation of new features
4. The market is saturated with one or two open source applications that everyone uses for each sector

Question: In such a world it would not make sense to make new applications from scratch by using spec driven development because working applications already exist. The only use case for spec driven development would be to improve existing applications or change the human user interface of an app.

Where You're Correct

You're right that greenfield development becomes less common. If one or two dominant open-source applications already serve each sector well, most developers would indeed contribute to those existing projects rather than building redundant competitors. The network effects and community momentum around established projects would make starting from scratch economically irrational for most scenarios.

Where You're Missing Use Cases

Niche and Adjacent Markets

Specialized subdomains still need new applications. Even if a dominant general-purpose accounting app exists, there might not be one optimized for a specific industry (e.g., sustainable forestry accounting, or crypto-native accounting). Spec-driven development would still be the fastest way to build these specialized tools. The market isn't completely saturated—it's saturated at the top level, but niches remain.

Architectural Rethinking

Fundamental reimagination of how a problem is solved. Sometimes the existing dominant app was built with assumptions that no longer hold true. Maybe the sector's needs shifted (remote-first workflows, mobile-first, AI-native workflows, etc.). Rather than incrementally modifying the existing app's architecture, building a new application with a different foundational design might be the cleaner path. Spec-driven development would enable this cleanly.

Interoperability and Ecosystem Tools

New tools that *connect to* dominant applications. Not every useful application needs to replace an existing one. You might build new applications that integrate with the dominant tool—plugins, adapters, data transformation layers, or complementary services. A spec-driven approach could define how these integrate before implementation.

Migration and Transition Tools

Applications specifically designed to help users migrate from the dominant tool. These are purpose-built for a temporary need but still valuable to develop using specs.

Regional/Language-Specific Variants

Localized versions with cultural or regulatory adaptations. A dominant app might exist globally, but a region with unique regulatory requirements (data residency laws, accounting standards, language-specific UX) might benefit from a spec-driven rebuild tailored to that context.

Experimentation and Innovation Cycles

Testing radical new ideas before integrating them upstream. A developer might use spec-driven development to prototype a fundamentally different UI paradigm or workflow optimization, intentionally *separate* from the dominant project. If it proves valuable, it could later be adopted or merged back.

The Deeper Point You're Missing

Spec-driven development isn't just a technique for *what* to build—it's a communication and coordination tool. Even when improving existing applications, specs serve crucial purposes: they let distributed human contributors understand *what* to build before they code, they reduce rework, and they enable async collaboration. So yes, you're right that spec-driven development would shift focus toward incremental improvement, but it would still be used heavily in that context because it's *how* you coordinate that improvement at scale.

Your core insight is sharp, though: the venture-scale startup model of "build an app and hope it dominates" would largely disappear, and most development would become maintenance, improvement, and niche specialization of existing projects.

Does this address gaps in your thinking, or are you skeptical of some of these use cases?

Revision #2
Created 2026-05-31 10:46:35 UTC by dpasn
Updated 2026-05-31 10:47:39 UTC by dpasn