Software customization strategies

There are many examples of software products that can be customized in many ways. I think in general people think that giving users too many choices will inevitably reduce the quality of the product. In contrast, a strongly constrained ecosystem such as Apple for example, will benefit from constant, highest quality reviews, and therefore no poor quality apps or user experience. Is there a middle ground, perhaps one where the company invests in certain community generated ideas and brings them to high quality standards? Has there ever been a case where a company paid some of the community authors to work on and improve their custom products?

This dilemma between providing too much flexibility for users and maintaining tight control for quality—has been a long-standing issue in software product design. Historically, software ecosystems have tended to lean toward either extreme: open, customizable platforms (e.g., Android or WordPress) versus more closed, highly curated environments (e.g., Apple’s App Store).

However, there are examples of companies finding a middle ground, where they balance flexibility with quality by investing in community-generated content, sometimes even paying contributors to bring their ideas up to a professional standard. Here are some cases where companies have done this:

**1. Valve (Steam Workshop)**

* **Approach**: Steam Workshop allows users to create mods and custom content for games. For some titles, Valve has taken high-quality mods and integrated them into the core game or sold them as paid downloadable content (DLC), giving creators a share of the revenue.
* **Monetization**: Valve has a history of paying community members to refine and bring certain mods up to commercial quality (e.g., *Counter-Strike* and *Dota* were originally mods).
* **Outcome**: This approach fosters a thriving community and keeps the platform flexible, while giving Valve control over which community-generated content meets their standards.

**2. Epic Games (Fortnite Creative)**

* **Approach**: In *Fortnite*, Epic Games has opened up creative tools for the community, allowing players to create their own game modes and maps. Some of the best user-generated content has been featured prominently in the game.
* **Monetization**: Epic has paid creators for their content through its “Support-A-Creator” program, where creators earn money when players spend in-game currency on their maps.
* **Outcome**: This hybrid approach allows Epic to maintain a highly polished core game while giving users freedom to experiment, with financial rewards for standout community content.

**3. Red Hat and Open Source Software**

* **Approach**: Red Hat takes open-source community projects (such as Linux distributions) and polishes them for enterprise use, providing high-quality, stable releases. Red Hat also pays open-source contributors and maintains a close relationship with the community while upholding commercial standards.
* **Monetization**: While the software is free, Red Hat sells support and professional services around the open-source projects.
* **Outcome**: This strategy keeps the ecosystem open and vibrant while maintaining high-quality, enterprise-ready solutions for customers.

**4. Mozilla (Firefox Extensions)**

* **Approach**: Mozilla encourages users to develop Firefox extensions and themes but maintains strict quality control through its add-on review process. The company occasionally partners with community developers to refine and promote particularly useful or innovative extensions.
* **Monetization**: Although Mozilla does not usually pay developers directly, they do support the development of certain extensions through grants and funds, ensuring some community-generated ideas meet a high-quality standard.
* **Outcome**: This strikes a balance between extensibility for users and a curated, reliable experience for the browser.

**5. Shopify (App Store)**

* **Approach**: Shopify’s App Store is open to third-party developers who can create apps that extend the functionality of the core Shopify platform. Shopify ensures that high-quality apps are featured and promoted, but does not entirely close off the platform to new ideas.
* **Monetization**: Shopify developers can earn substantial revenue from successful apps, and Shopify invests in community partnerships by offering developer tools, documentation, and support to help bring the best ideas to fruition.
* **Outcome**: This allows Shopify to maintain a well-curated marketplace while encouraging innovation from its community of developers.

**Lessons from These Examples**

* **Curated Openness**: In these cases, companies create systems where the community is free to experiment, but only the highest-quality contributions are integrated into the main product or marketplace. This can be a win-win: users have a degree of flexibility, but the end-product maintains quality standards.
* **Revenue Sharing/Compensation**: Paying or otherwise rewarding community members for their best ideas ensures that contributors are motivated to produce high-quality work. It can be an effective way for companies to benefit from community creativity while maintaining professional standards.
* **Investment in Refining Ideas**: Companies often invest in refining good community-generated ideas, bringing them to a professional standard. This helps ensure that the final product is polished and aligns with brand quality, while still leveraging the innovation and diversity of community contributions.

**Middle-Ground Strategy for SaaS or Cloud Platforms**

In your own context—developing a platform that connects different document versions and allows for third-party components—you might explore a similar “middle-ground” strategy:

* **Create a Developer Program**: Allow third-party contributors to submit modules or integrations, but maintain control over which ones are promoted or made available in the main product. You could provide incentives (e.g., financial rewards, platform promotion) for the best extensions.
* **Quality Control Layer**: Introduce a review process where your company helps refine the most promising third-party contributions before integrating them into the core platform, maintaining a high-quality user experience.
* **Sponsored Community Projects**: Invest in community-generated ideas that align with your vision, offering grants or direct compensation to improve and polish these ideas for widespread use.

This approach leverages community innovation while ensuring a consistent, professional product experience.

## Potential downsides

While developer programs and quality control layers offer many benefits, they also come with potential downsides. These challenges can affect both the developer community and the company’s internal stakeholders, including shareholders. Below are some key issues that have arisen with such initiatives, along with examples of backlash and lessons learned:

**1. Community Frustration with Overregulation or Bureaucracy**

* **Problem**: Developers in an open ecosystem may feel stifled if the review process or quality control layers are too strict, slow, or opaque. The balance between quality assurance and flexibility can become a pain point.
* **Example**: In the early days of the **Apple App Store**, many developers expressed frustration over the length of time it took for apps to be reviewed and approved. Even today, there are complaints about inconsistency in how app rules are enforced.
* **Backlash**: Developers may feel their creativity is being hindered, or they might abandon the platform for less restrictive alternatives. A frustrated developer community can lead to fewer contributions and less innovation, which ultimately hurts the platform.
* **Lesson**: Transparency and clear communication about review criteria, timelines, and reasons for rejection are crucial to keeping developers engaged and motivated. Offering fast feedback loops and predictable processes can help mitigate backlash.

**2. Developer Fragmentation or Favoritism Perception**

* **Problem**: When companies sponsor or pay only certain community members or contributions, this can create a sense of favoritism. Some developers may feel marginalized if they aren’t among the selected few, leading to fragmentation in the community.
* **Example**: When **Epic Games** introduced its “Support-A-Creator” program, some developers felt that it favored popular creators with larger audiences rather than fostering innovation and quality. Similarly, when platforms like **GitHub** began offering sponsors to open-source developers, some contributors felt left out of the economic benefits.
* **Backlash**: A sense of favoritism can erode trust between the company and its broader community of developers. If developers feel that certain contributors are being given preferential treatment, it can lead to discord and even the loss of talented community members.
* **Lesson**: Transparent, merit-based selection processes and clear criteria for funding or sponsorship can help avoid this perception. Offering open, competitive programs where developers can apply for funding or support, with fair evaluation, can make a difference.

**3. Financial Investment Without Immediate Return**

* **Problem**: Investing in quality control layers and developer programs requires resources, which can affect profitability in the short term. If shareholders or company executives don’t see a quick return on investment, this can lead to pressure to cut back on such initiatives.
* **Example**: **Mozilla** has faced financial struggles over the years, partly due to its investment in community projects like Firefox extensions and its open-source mission. At times, this has raised concerns among stakeholders about whether the investment is paying off, particularly as browser market share declined.
* **Backlash**: Shareholders might push back if they feel that resources allocated to community-driven initiatives aren’t yielding direct financial gains. This can create internal tensions about priorities.
* **Lesson**: It’s important to clearly align developer programs with the company’s broader strategic goals. Communicating how these initiatives drive long-term value—whether through user engagement, innovation, or brand loyalty—can help justify the upfront costs to stakeholders.

**4. Dilution of Core Product Identity**

* **Problem**: By allowing external developers to contribute, there’s a risk that the core identity of the product could be diluted if the third-party contributions don’t align with the company’s vision or values. Even with a quality control layer, maintaining a cohesive product experience can be difficult.
* **Example**: Platforms like **Android** and **WordPress** have sometimes struggled with fragmentation due to the sheer number of third-party extensions, leading to inconsistent user experiences. This has hurt the platforms’ perception of reliability and cohesiveness in the eyes of some users.
* **Backlash**: If the ecosystem becomes too bloated or disjointed, it can negatively impact the overall user experience, causing customers to lose trust in the product. This can also confuse new users who may be overwhelmed by the range of options or face compatibility issues.
* **Lesson**: Striking a balance between openness and consistency is key. Offering clear guidelines and curating third-party contributions ensures that external content complements the core product rather than detracting from it.

**5. Security and Quality Risks**

* **Problem**: Introducing third-party components into a product increases the risk of security vulnerabilities, bugs, or performance issues. Even with a quality control layer, malicious or poorly coded contributions can slip through, leading to potential breaches or a degraded user experience.
* **Example**: **WordPress** has been criticized for security issues related to its vast ecosystem of third-party plugins, some of which are poorly maintained or insecure. Similarly, the **Android** platform has struggled with malware infiltrating third-party apps, despite Google’s efforts to improve app review processes.
* **Backlash**: Security breaches or widespread performance issues can damage the platform’s reputation, potentially leading to user attrition. Customers and partners may question the platform’s reliability, and the cost of addressing security issues can be significant.
* **Lesson**: Invest heavily in robust vetting and security practices, such as automated testing, code audits, and sandboxing. It’s also important to have mechanisms for quickly identifying and addressing vulnerabilities.

**6. Platform Dependency**

* **Problem**: If a platform becomes heavily dependent on third-party developers for innovation and new features, the company may lose control over its own roadmap. Relying too much on external contributions can create misalignment between the company’s core goals and what developers prioritize.
* **Example**: **Salesforce** has a robust ecosystem of third-party apps and integrations, but at times, the ecosystem’s growth has led to customer confusion about which apps are officially supported by Salesforce versus those that are third-party. This can create dependency risks and strategic misalignment.
* **Backlash**: Over-dependence on external contributors for core functionality can make the platform vulnerable to shifts in developer interest or community priorities, which may not always align with the company’s vision.
* **Lesson**: The company should continue investing in its own core development team to build key features and drive the product’s strategic vision, ensuring that third-party contributions are complementary rather than critical.

**Mitigating Backlash and Challenges**

To avoid the downsides of developer programs and quality control investments, companies can take several steps:

* **Foster a transparent and inclusive community culture**: Clearly communicate the purpose, goals, and benefits of the program. Be transparent about how contributions are selected and rewarded.
* **Balance control and freedom**: Allow room for creativity while maintaining a clear quality standard. Offering open APIs, but requiring certain security or performance metrics, can balance flexibility and reliability.
* **Measure and communicate long-term benefits**: Show how community-driven initiatives contribute to user engagement, retention, or innovation. Regularly communicate these successes to both the community and stakeholders.
* **Keep internal innovation alive**: Continue investing in internal R&D to maintain leadership over the platform’s core vision, avoiding over-reliance on third-party developers for critical features.

Finding the right balance is key—offering community developers creative freedom while maintaining strong quality control, security standards, and alignment with the company’s strategic goals.