# Generative UI Alone Won't Make Software Malleable

BRYAN MIN, University of California San Diego, USA

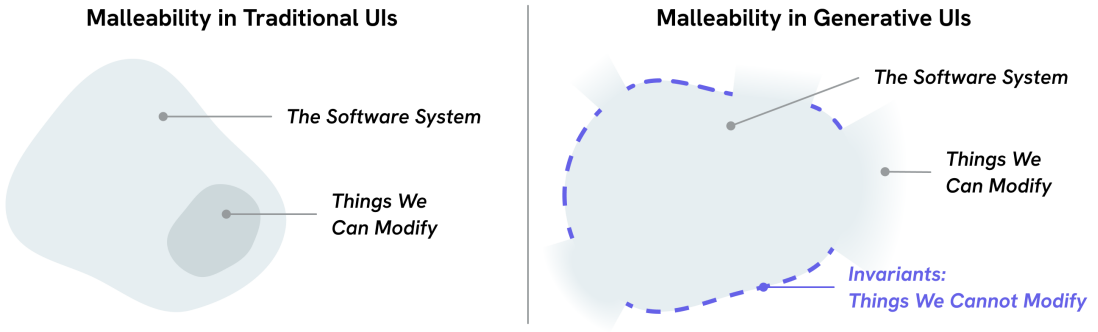


Fig. 1. While traditional user interfaces expose a limited subset of software features that users can modify through customization, generative UIs make nearly all aspects of an interface modifiable and ultimately provide too little structure for users to understand what can even be customized. To move generative UIs toward the vision of malleable software, systems must support the discovery of interface *invariants* and explicitly design these invariants into software to provide structure for customization. Doing so requires the HCI community to identify which aspects of software interfaces must remain stable as users modify them.

CCS Concepts: • **Human-centered computing** → **HCI design and evaluation methods**; **Interaction paradigms**.

Additional Key Words and Phrases: Generative UI, Malleable Software

## 1 Introduction

Generative user interfaces (GenUIs) are rapidly expanding what end-users can create and modify in software. This progress has demonstrated impressive potential toward achieving the longstanding vision of software becoming truly *malleable*, where users can flexibly modify interfaces to adapt them to their own tasks, workflows, and evolving needs [6, 7]. However, I argue that today's GenUI implementations alone will not crack the code to achieving malleable software. Specifically, the vision of malleable software will not emerge solely by improving our AI models.

I draw from Christopher Alexander's position who claims that enduring designs preserve ***invariants***—properties of a design that remain coherent as forms vary [1]. Alexander further argues that these invariants are not fully specified in advance. Instead, they are discovered through use, observation, and experience. In software, invariants may point to various UI components, functionalities, interaction techniques, or arrangements, and explicitly designing these invariants into software can provide the right guidance for users to productively make diverse modifications to their software. However, we have yet to identify the right set of invariants for GenUI systems. While current GenUI systems do allow users to vary the forms of software interfaces, they either restrict the system to a small set of predefined customizations or loosen the system to allow too much flexibility.

To achieve malleable software through GenUI, I call for the HCI community to foster better opportunities to discover invariants for GenUI. Specifically, I urge HCI to build *customization-ready* systems that prioritizes observing usage patterns over deploying stable software, observe common patterns across diverse software architectures that are deployed today, and boldly propose new *invariants* to design into GenUI systems.

Author's Contact Information: Bryan Min, bdmin@ucsd.edu, University of California San Diego, La Jolla, California, USA.

## 2 Why Current GenUI Systems Will Not Achieve Malleability

Malleable software envisions both ***expressiveness*** and ***ease*** of modification. Users should be able to make a wide range of changes to the software interface, but also be able to intuitively understand what can be changed and how to change it. Expressiveness alone is not sufficient if a modification is not obvious, and ease alone is insufficient if users are constrained to a narrow set of predefined options. Current GenUI implementations do not meet this bar and instead tend to fall into one of two cases.

*Structured Yet Inexpressive.* The first consists of **specification-based** GenUI systems, which define a fixed set of parameters or schemas that users can modify, often in the form of JSON [2, 3, 8, 9]. These systems make customization discoverable and relatively easy to use, as users are explicitly shown what aspects of the interface can be changed. However, they also significantly limit expressiveness. The space of possible modifications is constrained by what designers anticipated in advance—this is no different to the traditional approach of designing customization features case by case.

*Expressive Yet Unstable.* The second consists of **code-based** GenUI systems, which allow users to generate or modify interface code directly [4, 5, 10]. In principle, this approach is highly expressive, enabling users to create nearly any interface they can describe. In practice, however, these systems provide too little structure to support ease of modification. Users lack clear guidance about what parts of the generated interface are meaningful to change, what parts are safe to modify, or what parts should remain stable. As a result, making customizations in such systems grow difficult to achieve, as the interface does not effectively communicate how users should modify their software.

## 3 Fostering the Discovery of Invariants

If invariants are essential to making GenUI systems malleable, a central question is how these invariants can be discovered in practice. As an HCI community, we have both the power and the responsibility to shape the kinds of investigations we advocate for and value. I propose two ways that we, as researchers and reviewers, can foster the discovery of invariants for GenUI systems.

*Prioritizing Customizability Over Software Stability in HCI Systems.* Today's software systems are poorly suited for studying malleability, as businesses need to prioritize stability and safety over the risks associated with extensive customization. Consequently, these systems are *too safe* to support customization and ultimately suppresses the opportunity to discover invariants of GenUI. However, the HCI community offers a unique testbed for building alternative software systems that relax these constraints. Rather than concerning ourselves over production-ready software, we should open the community to *customization-ready* software that is primarily designed to enable large-scale, freeform customization in practice, even if this means deliberately relaxing conventional stability and security constraints. By creating software that is explicitly customization-ready, we can observe how users modify, extend, and reinterpret interfaces and help inform what parts of the software to prevent customization in GenUI systems.

*Case Studies of Software Engineering and Design Practices.* Important sources of insight about invariants for GenUI may lie in the practices of software engineers and designers working within domain-specific applications. Interviews and observational studies can reveal which parts of a system are treated as stable, which parts are routinely changed, and which modifications tend to cause breakdowns. Through case studies, we can abstract from these observations to identify recurring patterns and common software components that suggest the need for invariants.

## References


[1] Christopher Alexander. 1979. *The Timeless Way of Building*. Center for Environmental Structure, Vol. 1. Oxford University Press, New York.

[2] Google Developers Blog. 2025. Introducing A2UI: An Open Project for Agent-Driven Interfaces. https://developers.googleblog.com/introducing-a2ui-an-open-project-for-agent-driven-interfaces/. Published: December 15, 2025. Accessed: February 3, 2026.

[3] Yining Cao, Peiling Jiang, and Haijun Xia. 2025. Generative and Malleable User Interfaces with Evolving Task-Driven Data Schema. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '25)*. Association for Computing Machinery, New York, NY, USA, 21 pages. doi:10.1145/3706598.3713285

[4] Ruijia Cheng, Titus Barik, Alan Leung, Fred Hohman, and Jeffrey Nichols. 2024. BISCUIT: Scaffolding LLM-generated code with ephemeral UIs in computational notebooks. In *2024 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 13–23.

[5] Google Labs. 2026. Disco. https://labs.google/disco. Accessed: February 3, 2026.

[6] Geoffrey Litt, Josh Horowitz, Peter van Hardenberg, and Todd Matthews. 2025. Malleable Software: Restoring User Agency in a World of Locked-Down Apps. https://www.inkandswitch.com/essay/malleable-software/.

[7] Bryan Min, Allen Chen, Yining Cao, and Haijun Xia. 2025. Malleable Overview-Detail Interfaces. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '25)*. Association for Computing Machinery, New York, NY, USA, 26 pages. doi:10.1145/3706598.3714164

[8] Bryan Min and Haijun Xia. 2025. Meridian: A Design Framework for Malleable Overview-Detail Interfaces. In *Proceedings of the 38th Annual ACM Symposium on User Interface Software and Technology (UIST '25)*. Association for Computing Machinery, New York, NY, USA, Article 200, 14 pages. doi:10.1145/3746059.3747654

[9] OpenAI. 2025. Introducing Apps in ChatGPT. https://openai.com/index/introducing-apps-in-chatgpt/. Accessed January 2026.

[10] Priyan Vaithilingam, Elena L. Glassman, Jeevana Priya Inala, and Chenglong Wang. 2024. DynaVis: Dynamically Synthesized UI Widgets for Visualization Editing. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '24)*. Association for Computing Machinery, New York, NY, USA, Article 985, 17 pages. doi:10.1145/3613904.3642639