

Rigidity captures how much the user can shape the rules. Four values span a spectrum from system control to user control. A *fixed* rule is immutable: the email schema in Outlook requires a sender, a recipient, a subject, and a body, and no other rule can modify this. A *negotiable* rule can be relaxed within bounds but not redefined: the volume limiter on iOS lets the user adjust the maximum level within a fixed range, but not redefine how limiting works. A *malleable* rule can be redefined entirely by the user, like a keyboard shortcut in Photoshop that can be reassigned to any function. An *authorable* rule goes further: the user can create rules that did not previously exist, as when a user writes a macro in Excel or defines custom geometric constraints in Fusion 360.

Enforcement captures how much the user is bound by the rules during interaction. Four values span a spectrum from unyielding to yielding. A *persistent* rule does not yield: a minimum panel width blocks hard at its limit. An *elastic* rule yields under pressure but restores itself: the magnetic timeline in Final Cut Pro resists gaps between clips, but clips can be forced apart before re-snapping on release. An *escapable* rule tolerates bypass that leaves a lasting result: indentation in VS Code enforces a tab size, but manual spacing can override it locally and the override persists. A *liftable* rule can be removed from enforcement entirely: spell-check in Word can be disabled for a document.

These two dimensions are orthogonal: any rigidity value can combine with any enforcement value. Desktop spreadsheet grids are (*fixed, escapable*): the grid logic itself is immutable, but paste-special operations and direct in-cell editing bypass dependencies and the results persist. The macOS Dock is (*negotiable, elastic*): size is adjustable within bounds, and magnification on hover yields then restores. The state couple thus generates a design space of 16 combinations.

3 Implications for Generative UI

The framework has three direct implications for genUI design.

GenUI must generate structural behavior, not just structural form. Current generative approaches focus on producing interface layouts and components [4]. The framework reveals that this is insufficient: for every rule a system generates, it must also specify that rule's rigidity and enforcement. A genUI system that deposits fully-formed content into a (*fixed, persistent*) structure produces an artifact users cannot modify or escape, which is consistent with the difficulties Cao et al. [3] observed when participants struggled to revise and explore within generated structures. One that generates (*negotiable, liftable*) structure would instead let users adjust generated constraints within bounds and selectively suspend them while exploring. The difference between a rigid, opaque genUI and a flexible, steerable one is not a matter of "how much AI" but of which state couples the system assigns to its generated rules.

Consider a genUI system that generates a data dashboard. If the column layout rules are (*fixed, persistent*), the user cannot rearrange columns or break out of the generated grid. If they are (*malleable, escapable*), the user can redefine the layout entirely, and local overrides (manually resizing a column, dragging a widget outside the grid) persist as lasting deviations. The structural form is the same; the structural behavior is entirely different.

Structural states should align with workflow phases. Creative and analytical workflows move through phases with different structural needs. Exploration benefits from lower rigidity (*negotiable* or *malleable*) with *liftable* enforcement: the user can adjust or redefine constraints and suspend those that get in the way. Execution benefits from higher rigidity (*fixed*) with *persistent* enforcement: constraints hold firm so that the user can focus on content without structural interference.

A genUI system aware of workflow phase could dynamically shift rule states rather than imposing a single structural configuration throughout. During ideation, layout constraints might be (*negotiable, liftable*): adjustable and removable. As the user commits to a direction, the system could tighten to (*fixed, elastic*): the layout becomes fixed but still yields elastically under pressure, providing guidance without rigidity. During finalization, (*fixed, persistent*) locks the structure. This progression from loose to tight structure mirrors how designers naturally work, but current genUI systems have no vocabulary to express it.

The framework provides a shared specification language. For genUI to be steerable, designers and users need a vocabulary to communicate structural intent to the system. The state couple (rigidity, enforcement) provides a compact, domain-independent notation. A designer could specify that a generated layout's column constraints should be (*negotiable, elastic*) and its typography rules (*fixed, persistent*), giving the system precise behavioral targets rather than vague directives like "make it flexible." A user could request that generated constraints be "liftable" without knowing anything about the framework, and the system could interpret this as a shift along the enforcement dimension. The vocabulary scales from expert specification to natural-language interaction.

4 Discussion and Conclusion

Structural Interaction suggests that HCI and design practice will need to evolve in at least two ways to support genUI.

First, the unit of design must expand. Current practice centers on designing interactions with content: selecting, dragging, editing domain objects. When structure is generated rather than hardcoded, designers must also specify how each structural rule should respond to user action. This requires new design tools: notation systems for expressing structural states, simulation environments for previewing how a configuration behaves under user pressure, and evaluation methods that assess structural behavior independently of content quality.

Second, the relationship between user and structure must be renegotiated. In traditional interfaces, structure is the designer's province: users interact through structure, not with it. GenUI dissolves this boundary. If an LLM can generate and evolve structural rules [4], users need mechanisms to inspect, override, and author those rules, shifting along the rigidity dimension from *fixed* toward *authorable*. The framework clarifies that "user control over AI-generated interfaces" is not a single slider but at least two independent parameters per rule. A user who can override a generated constraint (*escapable*) but not redefine it (*fixed*) has a fundamentally different relationship to the system than one who can redefine constraints freely (*malleable*) but cannot escape them during interaction (*persistent*).

These claims require empirical validation. Other modalities or domains may reveal additional values or new dimensions. However, we believe the core argument is sound: as interfaces become generative, the organizational schemas through which users act become a primary design material. Structure is not merely something interfaces have; it is something interfaces do, and designing genUI through structural behavior opens new possibilities for interfaces that are simultaneously powerful, flexible, and human-centered.

References

- [1] Michel Beaudouin-Lafon. 2000. Instrumental interaction: an interaction model for designing post-WIMP user interfaces. In *Proc. CHI '00*. ACM, 446–453.
- [2] Joanna Bergström and Kasper Hornbæk. 2025. DIRA: A model of the user interface. *Int. J. Hum.-Comput. Stud.* 193, C (2025), 14 pages.
- [3] Yining Cao, Yiyi Huang, Anh Truong, Hijung Valentina Shin, and Haijun Xia. 2025. Compositional Structures as Substrates for Human-AI Co-creation Environment. In *Proc. CHI '25*. ACM, Article 188.
- [4] Yining Cao, Peiling Jiang, and Haijun Xia. 2025. Generative and Malleable User Interfaces with Generative and Evolving Task-Driven Data Model. In *Proc. CHI '25*. ACM, Article 686.
- [5] Vincent Cavez, Caroline Appert, and Emmanuel Pietriga. 2024. Spreadsheets on Interactive Surfaces: Breaking through the Grid with the Pen. *ACM Trans. Comput.-Hum. Interact.* 31, 2, Article 16, 33 pages.
- [6] Clemens N. Klokrose, James R. Eagan, Siemen Baader, Wendy Mackay, and Michel Beaudouin-Lafon. 2015. Webstrates: Shareable Dynamic Media. In *Proc. UIST '15*. ACM, 280–290.
- [7] Wendy E. Mackay and Michel Beaudouin-Lafon. 2025. Interaction Substrates: Combining Power and Simplicity in Interactive Systems. In *Proc. CHI '25*. ACM, Article 687.
- [8] Philip Tchernavskij. 2019. *Designing and Programming Malleable Software*. Ph.D. Dissertation. Université Paris-Saclay.
- [9] Jenifer Tidwell. 2010. *Designing Interfaces*. O'Reilly Media.
- [10] Vincent Cavez, Kashif Imteyaz, and Anne-Flore Cabouat. 2026. Structural Interaction: Shifting the Focus of User Interface Design. Under review.

Unpublished working draft.
Not for distribution.