Chapter 7

Expanding Access to Computational Design:
Integrating Agentic AI and Web3 to Advance Node-Based AEC Workflows

Carlo Beltracchi
https://orcid.org/0009-0003-3585-4612
Politecnico di Milano, Italy

Ahmed Elmaraghy
https://orcid.org/0000-0002-0181-0641
Politecnico di Milano, Italy

Pierpaolo Ruttico
https://orcid.org/0000-0003-2642-6053
Politecnico di Milano, Italy

Simone Maccagnan
https://orcid.org/0009-0003-8330-9782
Gimac, Italy

ABSTRACT

This contribution proposes a way to broaden access to computational design by combining: (1) an agentic workflow where AI micro-agents translate natural-language prompts into executable, self-verified parametric graphs; (2) a data-driven economy in which each reuse of logic triggers automatic micropayments; and (3) a decentralised network that stores versions, rights and transactions on-chain. Assessor, provider and validator agents assemble, check and publish sub-graphs serialised as

DOI: 10.4018/979-8-3373-3256-7.ch007

semi-fungible tokens; a blockchain ledger tracks lineage and redistributes royalties. The platform merges open-source principles with Web3 incentives: newcomers gain ready-to-use solutions, experienced designers monetise know-how, and the community governs parameters via on-chain voting. Supported by robotic 3D-printing partners, the framework targets XR adoption: tokenised parametric graphs power virtual configurators for (1:1) design alternatives; users and curators vary parameters within constraints and record reuse on-chain, supporting an inclusive creator economy across the generative process.

FROM "LOW-CODE" PROMISE TO REAL-WORLD BARRIERS: WHY AI + WEB3 ARE NEEDED TO BROADEN ACCESS TO COMPUTATIONAL DESIGN

Over the past three decades digital practice has undergone a radical shift. Early computer-aided drafting merely transposed the drawing board onto a screen; today computational design asks the practitioner to script the logic that generates form. Visual-graph environments such as Grasshopper, Dynamo (Autodesk, n.d.) and Houdini - alongside Python - and JavaScript-based generative frameworks - now sit at the core of architectural, engineering and product-design workflows. Marketing material brands them "low-code": drag a few nodes, tweak some sliders and obtain intricate geometry without writing a single line.

Daily experience reveals a more demanding reality. The first barrier is *cognitive*: users must internalise data trees, functional recursions and flow-based reasoning foreign to traditional drafting. The second is *ecosystemic*: productivity hinges on a patchwork of third-party plug-ins whose documentation is uneven and whose versions frequently collide. The third is *technological-manufacturing*: anyone aiming for digital fabrication must account for machine tolerances, robot kinematics and material rheology -factors that reshape the very topology of the parametric graph.

These pressures generate a double gap. On the educational side, students and junior professionals must first master geometric modelling, then absorb algorithmic thinking, then overlay production knowledge; the learning curve is steep, and attrition remains higher than in conventional modelling courses (Aish & Hanna, 2017). On the corporate side, firms pour resources into training staff capable of maintaining functional definitions that, if left unattended, degrade quickly. The paradox is obvious: tools capable of solving complex geometry, optimising performance and enabling short-run customisation remain confined to circles of specialists.

In parallel, a more radical need emerges: to natively integrate user friendly tools within immersive (XR) environments—not as mere viewers of text-generated models, but as executable "fields of possibility": parametric algorithms that expose

Related Content