BLD2359

Feature, Add-In, or AI? Should You Buy, Build, or Hack in 2026?

Panel Summary

Carl Storms - Moderator
Newforma

Mostafa El Ayoubi - Panelist
Orkestra Online

Michael Kilkelly - Panelist
Ulteig

John Pierson - Panelist
Autodesk

Brodi Scott - Panelist
BIM One

Learning Objectives

- Assess when to rely on native platform features, add-ins, or AI-driven solutions based on real-world constraints
- Compare how different approaches (features, integrations, AI) perform across speed, risk, and long-term impact
- Evaluate expert responses to real-world scenarios to identify tradeoffs and decision strategies
- Determine the most appropriate solution path based on available resources, timelines, and risk tolerance

Description

Feature, Add-In, or AI? Should You Buy, Build, or Hack in 2026

AI is changing how work gets done in AECO, but it’s also complicating the path forward. Do you wait for a feature, bolt on an add-in, or hack together an AI solution?

In this panel, industry experts will respond to real-world scenarios and rapid-fire questions. Drawing from experience across platforms, add-ins, and AI workflows, they’ll weigh tradeoffs between features, integrations, and fast AI solutions, including what works, what breaks, and what creates risk.

This session is for teams working within real constraints. Whether you have development resources or you’re figuring it out on your own, you’ll leave better equipped to decide what to buy, what to build, and when a quick solution makes sense.

Panel Moderator





Carl Storms
Technical Success Manager | Newforma
AECO Tech Explorer | The BIMsider

I've been working in the AECO industry since 1998, spending much of that time as a proud BIM crusader helping teams adopt digital workflows and rethink how projects get delivered.

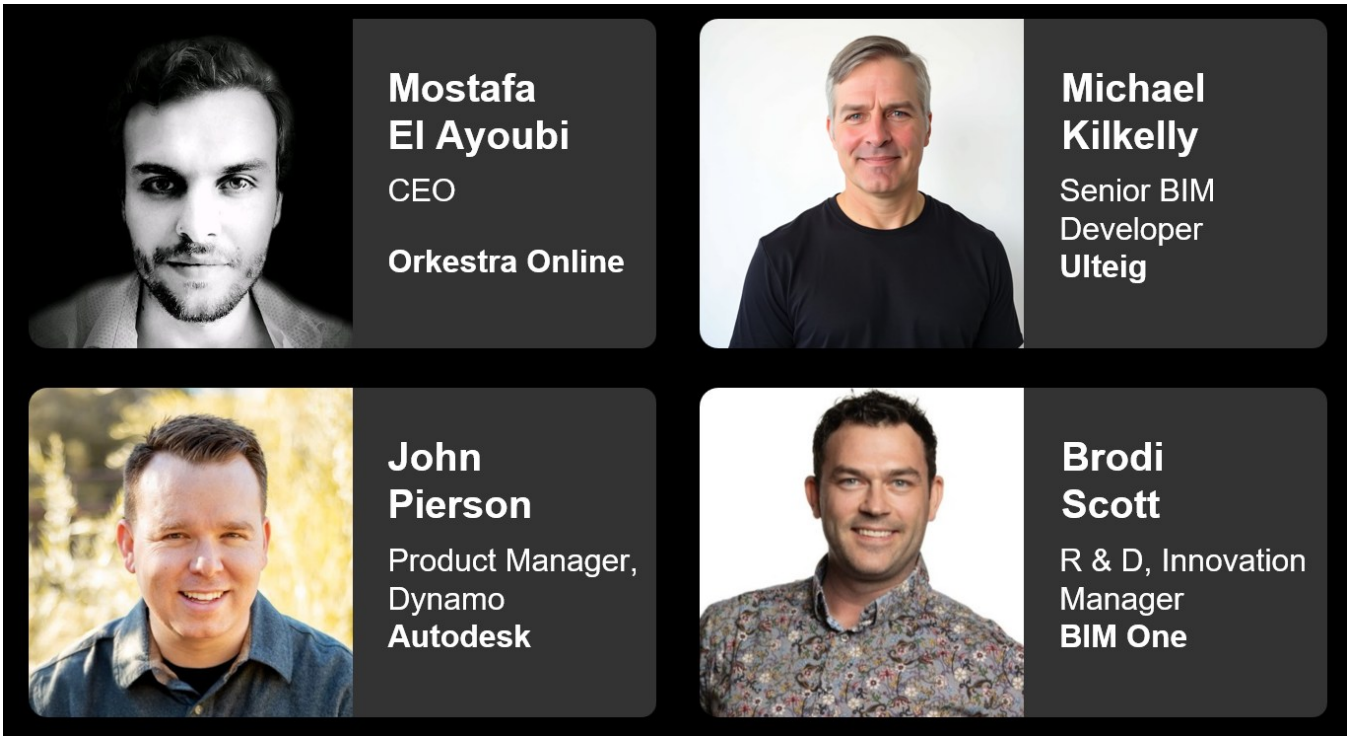
These days I'm more of an AECO tech explorer.

I like trying new tools, experimenting with ideas, and figuring out how emerging technology actually fits into real workflows. Lately that means spending a lot of time exploring AI, automation, APIs, and low-code or no-code tools, often experimenting with AI-assisted coding along the way. My goal is simple: explore the technology early, share what I learn, and help others decide if it's worth their time.

I share what I learn through The BIMsider, speaking at industry events, and conversations across the AECO community. Over the years I've also hosted or co-hosted several industry podcasts and have been fortunate to be part of the Autodesk Expert Elite community.

If you enjoy talking about technology, workflows, gadgets, or clever hacks that make work a little easier, we'll probably get along just fine.

The Panelists – The Important People



- Mostafa El Ayoubi
  - CEO | Orkestra Online
  - Co-Founder - Design Technology specialist | Data Shapes
- Michael Kilkelly
  - Senior BIM Developer | Ulteig
  - Founder | ArchSmarter
- John Pierson
  - Product Manager, Dynamo | Autodesk
  - Maker of okay things for the AEC industry | The Links
- Brodi Scott
  - Research, Development & Innovation Manager | BIM One

Panel Summary

This is a summary of the conversations, debates, knowledge sharing, and audience Q&A from the live panel at Autodesk University 2026.

A big thanks to John Pierson for his quick thinking and for recording the session using the AI tool "Voicenotes". Without that recording, a lot of the great conversation, questions, and ideas shared during the session would have been left on the AU session floor in Vegas.

Rapid Fire Questions

These quick-answer questions got the panelists warmed up before we jumped into the main topic.

- What is your AI platform of choice? John said Codex, Mostafa said OkPy and the rest said Claude Code

- MCP: Game changer or overhyped? (MCP = Model Context Protocol) There were 2 votes for overhyped, 1 for game changer, 1 vote for both game changer and overhyped and 1 for misunderstood.

- Where do you spend your AI time: Desktop app, Terminal, IDE, or somewhere else? Desktop app = 3, Terminal = 2, IDE = 1, Somewhere else = 2 (Web App & OkPy)

- What is one thing about AI that people are overestimating, or one thing they're underestimating?
  - Overestimated: "one-shot" deliverables and reliability of outputs.
  - Underestimated: current state-of-the-art capability, and the breadth/speed of R&D across industries.

Panel Discussion Questions

Now these are the questions that took us into the deeper discussion around "Feature, Add-In, or AI? Should You Buy, Build, or Hack in 2026?"

- When you need an AI capability that doesn't exist today, what's your first instinct: search for a tool, ask your vendor, build something yourself, or hack a solution together?
  - Common flow: try to research what exists first, then prototype/hack, then decide whether to harden.
  - "Throwaway vs committed" framing:
    - If it's disposable (quick bump/proof), hack fast (vibe coding).
    - If it's important/mission-critical, build deliberately and be prepared to maintain/support it.
    - If interest is low and maintenance burden is high, prefer existing paid solutions.

- Why the answer has changed over ~18 months:
  - Tooling has improved dramatically (scope and speed of building increased).
  - Comfort/"relationship" with tools matters; experience using AI tools improves outcomes.
  - Reusing prior repositories/codebases as references helps (style/structure guides reduce failure modes)

- How do you decide whether something should be a feature, an add-in, or a standalone AI solution?

  - Decision factors:
    - Where the workflow lives (e.g., inside **Revit** → more likely add-in).
    - Deterministic vs probabilistic needs:
      - Prefer deterministic execution for model-altering operations and repeatable production outputs.
      - AI is helpful to explore an ill-defined problem, then "compile" the outcome into deterministic code/scripts.
    - Cost, effort, and lifecycle:
      - "Building is easier now," but support, maintenance, and handling user complaints remain significant.
      - Technical ownership matters: only build what you're willing to maintain.
  - Vendor perspective (example from **Dynamo**):
    - ML can be embedded as predictable, trained-data features (e.g., node suggestion/autocomplete) without making every step probabilistic.
    - Vendors aim to move "fast responsibly," using curated/guardrailed experiences (e.g., controlled assistants) rather than unleashing unconstrained actions on models.

- AI has made it easier than ever to hack together solutions. When does a quick win become technical debt?

  - Signals a hack becomes debt:
    - When people forget it's a hack and start scaling/relying on it.
    - When many users depend on it and failure becomes operationally costly.
    - When it depends on upstream components you don't control (breaking changes create immediate debt).
  - QA expectations:
    - Review AI-generated code; don't accept confident outputs at face value.
    - Use layered review (including AI to review AI outputs) plus human checks.
    - Be mindful of edge cases (e.g., API quirks and uncommon model conditions).
  - Liability risk escalator:

- If outputs are not reviewed, this can cross from “irresponsible” to “reckless,” potentially affecting professional liability exposure and insurance defensibility (panel referenced an AI risk/legal session).
- Are software vendors moving fast enough with AI capabilities, or should users expect to build their own solutions on top of platforms?
    - Shared view:
        - Some users will always outpace vendors because their problems are immediate and specific.
        - Firms should expect to build more custom solutions as AI lowers the barrier for “citizen developers.”
        - The bottleneck shifts from building to governance: avoiding reinvention, ensuring quality, spreading value across the firm.
    - Platform requirements:
        - Strong APIs/SDKs and tool discoverability matter to prevent duplicated effort.
        - Lack of APIs/MCP-like connectivity may become a disqualifier for cross-platform workflows.

- Is the future of software buying features, or buying the ability to create your own features?
    - General direction:
        - Tools will increasingly sell extensibility: the capability to build/customize features.
        - “One size fits all” won’t meet firm-specific workflows; customization becomes essential.
    - Tension for vendors:
        - Extensibility empowers users but also raises the question of users recreating vendor functionality.
        - A stable “foundation + connectivity” strategy (like an app ecosystem) was favored over monolithic “perfect solutions.”

Audience Q&A highlights

Once the panel addressed all the prepared questions, we opened the floor for questions from the audience, and below are the highs from that Q&A session.

- MCP cost/benefit vs deterministic add-ins:
    - If you already have deterministic **Revit** add-ins that work, there may be no reason to wrap them as MCP tool calls.

  - Natural language can reduce the need for deep technical knowledge, but for expert users who already “know where to click,” typing may be less attractive.
- Open-source vs closed-source models:
  - Panelists saw value in exploring open models for fit/cost/control, but noted constraints (hardware needs, context windows, and self-hosting overhead).
  - Open models can be a fallback if API availability or market conditions change.
  - Open-source package maintainers raised the “who pays?” problem for embedding paid AI features; “bring your own key” is possible but adds UX/security complexity.
- Custom instructions, prompting, and workflow techniques:
  - Examples included style constraints, artifact-based explanations (HTML outputs), and forcing planning artifacts (plan.md).
  - Voice-to-text used by some to draft prompts; others prefer typing/iterating and having the model “interview” them.
  - Strategy: use stronger models to create plans, then cheaper models to execute.
- Justifying in-house tools vs buying (support skepticism):
  - Use AI to generate unit tests and strengthen QA evidence.
  - Improve documentation and publish internal repos so others can maintain/extend.
  - Start small and quantify ROI (time/money saved) to make the case.
  - If tools break, use AI to help troubleshoot, but still require internal capability and handover paths.
- Skills to focus on (2–5 years):
  - Don’t blindly accept outputs; learn from what AI generates.
  - Domain mastery remains critical: experts automate better because they can specify and verify.
  - Education still needs fundamentals (e.g., coding) to understand what’s happening “behind the scenes,” even if AI reduces daily need to write everything manually.
  - Human creative problem-solving and discernment (real vs AI-generated “slop”) remain differentiators.

In Conclusion

What this panel discussion has shown, as predicted, is that there isn't one right answer.

Sometimes the best solution is a feature that's already built. Sometimes it's an add-in. Sometimes it's a weekend project powered by AI. The goal isn't to use AI just for the sake of using AI. The goal is to solve real problems.

What also came from this panel is that the future needs to be considered when choosing between these options.

If you're waiting for a feature, you're also waiting for the support, polish, testing, and QA that goes into it. With an add-in, you're somewhere in the middle. It might come with support, but maybe it doesn't. It might be a one-off tool that fills a gap today, but what happens when it's no longer supported? Does that help or hurt the outcome you need?

Then there's the tool you hack together with AI for a quick win. If all you need is a quick, one-and-done solution, that can be a great option.

But if you want other people in your company to use it, or maybe you want to share it with the AECO community, then you need to think about its future. Who will maintain it? Will it still work six or 24 months from now? Does it only support the current version, or all the supported versions? For example, does it work with Revit 2027, 2026, 2025, 2024, and 2023? Can someone else understand how it works? And what happens when the AI-generated code needs fixing?

The quick hack might solve today's problem, but if it's going to become something bigger, you need to start thinking about tomorrow's problem too.

Whether you buy, build, or hack, the real advantage comes from the people asking better questions, choosing the right tools, and solving better problems.

As **Fei-Fei Li** said:

