AIA25

ID: WE119

Title: Energy Modeling on Autopilot: Scripting Efficient Architectural Designs

Date: Wednesday, June 4

Time: 11:00am - 12:00pm

LU: 1 HSW/RIBA/LU

Program Summary
[Session description as published on conference website]

Learning Objectives

1. Find out how programming scripts can automate energy modeling and analysis in architectural projects.
2. Learn how to integrate automated energy tools into existing design workflows for real-time feedback.
3. Review key areas where automated energy analysis can enhance building performance and reduce costs.
4. See how to apply best practices for using automation to achieve energy-efficient architectural designs.

Speakers:

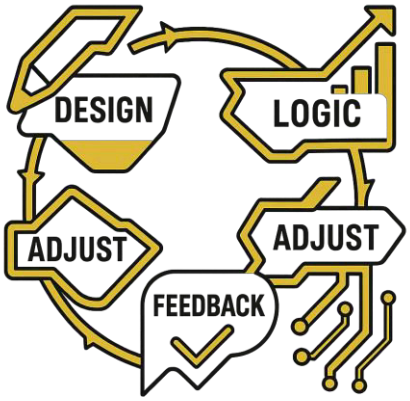
- Enrique Galicia Tovar

AIA25

Automate energy modeling

using scripts

1.Let's Be Clear: This Isn't Magic



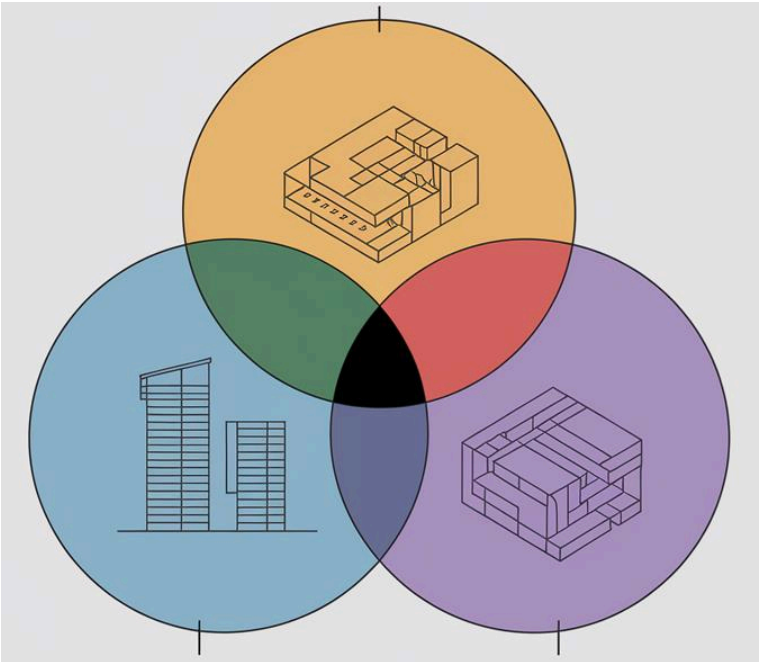
Let's start from a real place—this isn't magic, it's logic. The goal here isn't to replace what you do as a designer, but to help your design decisions talk back to you sooner. We'll use tools you probably already have—Forma, Revit, Dynamo—and combine them in a way that lets performance become visible while you're still designing, not after. You don't need to code, just to think in cause and effect. Inputs like room areas or window sizes go in. Visual feedback or tagged alerts come out. It's not about building simulations—it's about building rules that you'd usually apply manually, and turning them into something you can reuse across every project. We'll show you how to start with one script, one metric, and grow from there. You'll see how it works, how to adapt it, and where it saves time immediately. That's the only promise here: make your logic visible, reusable, and useful—before it's too late to change anything

2.What You'll Actually Learn

We'll work with Forma, Revit, and Dynamo. No niche platforms. These are common tools—probably already installed—just used in coordination. Forma gives us geometry and early-stage climate data. Revit brings in parameters. Dynamo connects them through logic.

3.The Tools We're Using

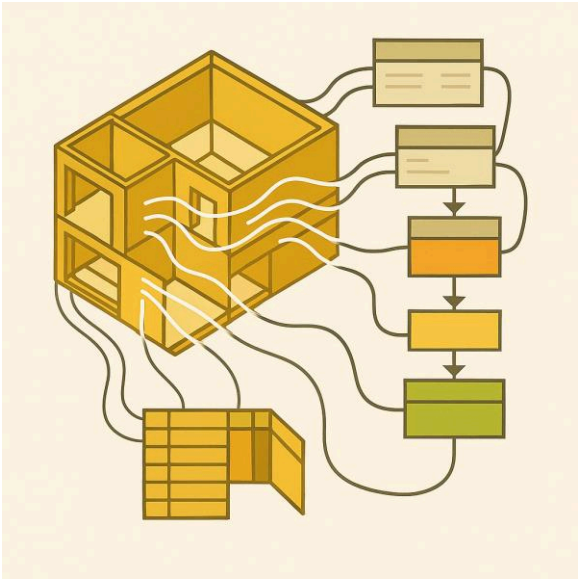
The workflow today is centered around tools you've either used or already have access to. Forma is for concept design and early insight. Revit is for documentation and geometry. Dynamo connects the logic between them. This stack is deliberate—not overly technical, not reliant on extra plug-ins. It's meant to show that with a small amount of structure and a few thoughtful scripts, you can take what you're already doing and get more feedback, earlier, without adding workload.



AIA25

4.What This Needs (and Doesn't)

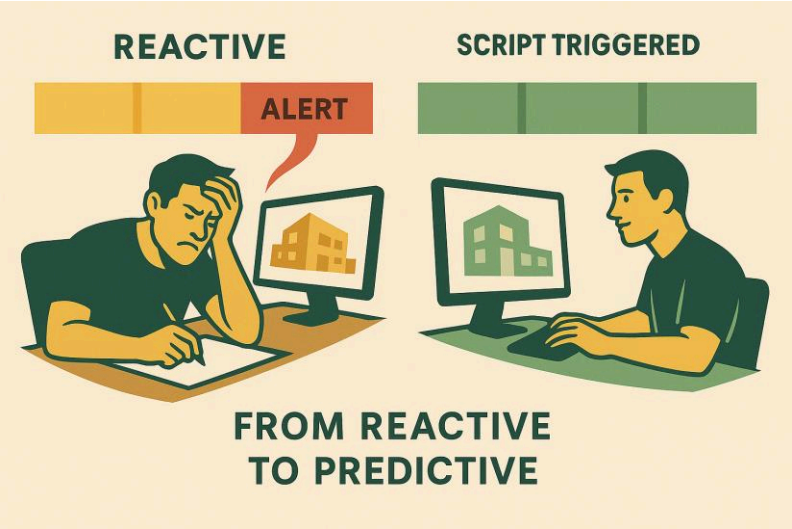
You don't need to be a programmer. You don't need to set up environments or write Python. This approach is built around using what you already model: solids, walls, floors, rooms, parameters. We're translating design questions into logic—and Dynamo just becomes the place where that logic lives. The key here isn't coding ability—it's clarity. If you can say, "If the glazing ratio is over 40%, show me," you already have the logic. The rest is structure.



5.The Real Workflow

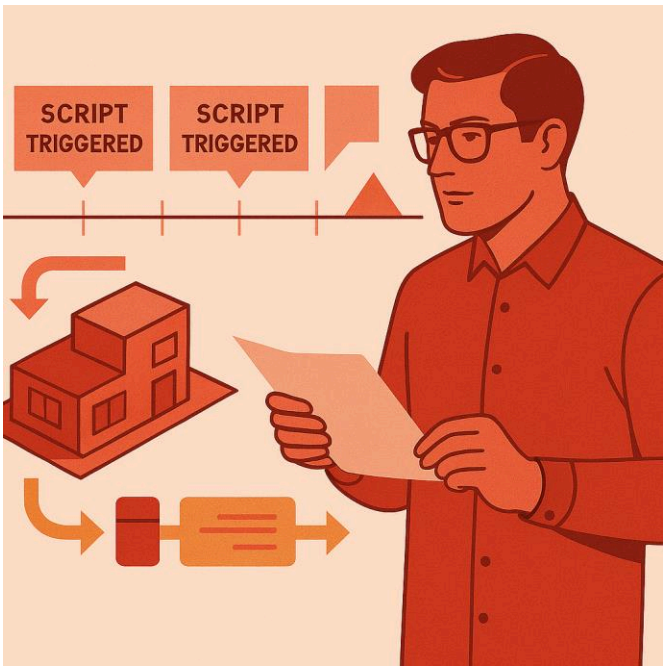
We're going to follow a very specific path. Geometry comes from Forma or Revit. Parameters are already there. A script—built in Dynamo—reads that information and applies a rule. The output might be a tag, a warning color, or a note on performance. Nothing leaves your modeling environment. You don't jump to simulation tools. You don't export and import. Everything happens inside your actual design flow—just with a bit of logic layered on top.

AIA25



6.Why Script at All?

Scripting matters because most performance feedback comes too late. You don't need to be told that—it's happened to all of us. You're in DD, you submit for review, and only then someone flags an energy issue you could've caught earlier. Scripts give you a way to build in those rules at the SD phase—or even in concept massing. That means faster approvals, fewer surprises, and better design with less rework. The earlier you get feedback, the less it costs.



7.What Scripts Can Control

You're not simulating full systems. You're scripting the kind of decisions you already make: Is this zone too deep? Is the glazing too high on this face? Is the room too tall for the mechanical system we're planning? These are logic-based conditions. You'd usually check them manually. With scripts, you just describe the condition once—and it checks itself from that point on. It's assistive logic, not automated design.

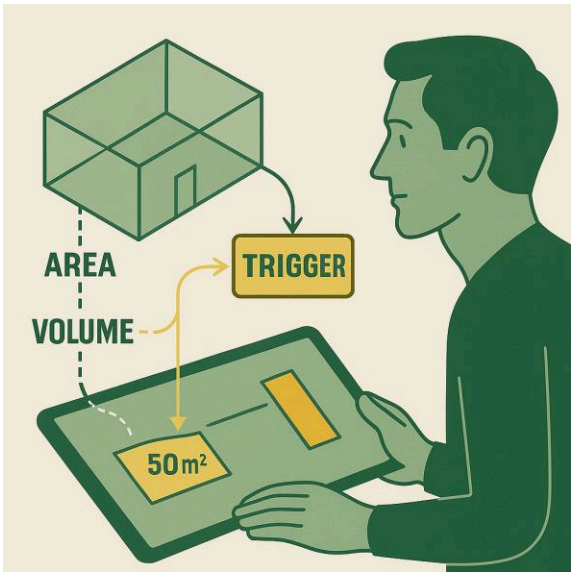
8.Simple Input Types

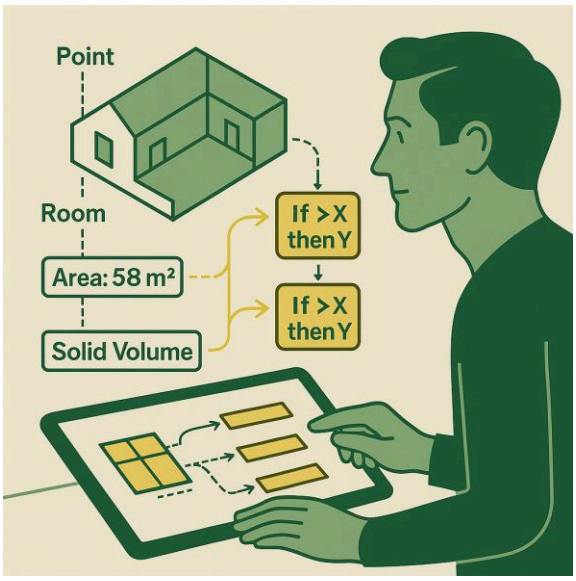
To be clear, we're not inventing inputs here. Scripts use what's already in your model: a point, a solid, a room boundary, a parameter like area or volume. These aren't new data types—you've worked with them for years. What changes is how we respond to them. Instead of seeing area as just a label, we see it as a trigger: "If the area is over 50 m², do this." That's how automation starts—by assigning meaning to something you already have.

9.Simple Output Types

The output from a script doesn't need to be fancy. It can be a color fill, a new parameter value, or a tag. It can even be structured data you send to a dashboard. The important thing is: it's visible. It helps you make better decisions. And it's automated. Once the script runs, it does the checking and marking for you. It doesn't replace your judgment—it just gives you more eyes on the model without more people.

AIA25





10.The Logic Is Repeatable

This is where scripting starts to pay off. Once you've built a logic path—say, a check on window-to-wall ratio—you can apply it on every project. You're not redoing the work, just connecting it to a new model. And since that script is based on parameters, not specific geometry, it scales automatically. The script becomes a knowledge object. It captures how you think—and lets your whole team benefit from it.

11.You Don't Need to Build from Scratch

You'll see examples today that you can copy and adapt. You don't need to wire a new graph or write logic from zero. Most of what works well in scripting is built on patterns—clear flows that can be duplicated and tuned. So instead of being overwhelmed by a blank canvas, you start with something that already functions. You tweak a rule, change a threshold, and it's ready for your project.

AIA25

12 – Let's Start With Basic Dynamo Logic

Before we get into metrics or simulations, we need to understand how automation actually begins: with structured thinking. In Dynamo, this means extracting the geometry already in your model, understanding the data attached to it, and applying logic that responds to design intent. This isn't coding—it's visual logic mapping. Whether you're pulling walls, tagging rooms, or coloring mismatches, every action starts with clean, modular graph logic. This slide introduces the foundation: geometry extraction, parameter management, and simple logical relationships. These are the DNA of every script that follows.

13 – Reading, Responding, and Applying

Dynamo reads the model, reacts to conditions, and returns results—all in real time. We show how it identifies the elements you care about (walls, rooms, windows), extracts their properties (area, function, height), and then uses that information to calculate or flag results. That result might be a color override, a parameter update, or even a tag placed on the view. And because the logic is modular, each piece of the script does one job: input, evaluate, output. This makes every automation transparent, reusable, and easy to edit—whether you're changing a rule or applying it to a different model.

14 – Reuse, Share, and Scale

Once a script works, it doesn't stay stuck in one model. Want to apply the same logic to a new building? Swap the input elements. Want to change the WWR threshold? Update one number. These scripts are built to be shared across projects and teams, using Dynamo Player to make execution simple—even for users who never open the graph. The result is consistency across deliverables, fewer manual checks, and better design outcomes. Structured logic scales. It doesn't just solve a problem—it solves it everywhere you need it to.

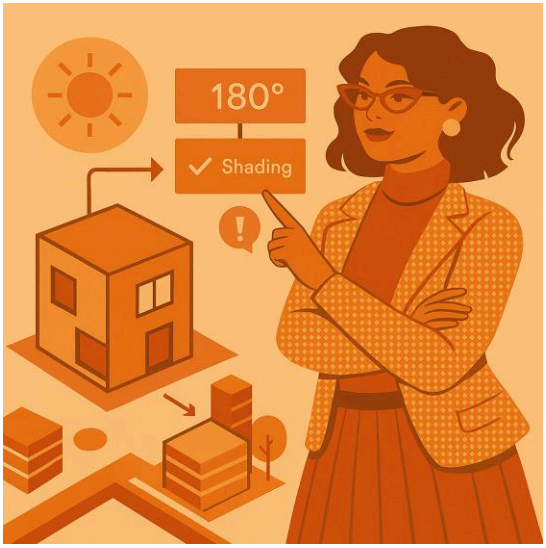
15 – Conceptual Massing: Orientation, Sun, and Wind Analysis

In the earliest stages of design, we don't have windows, walls, or systems—we only have form. This is where conceptual automation makes its mark. By extracting vector directions from mass faces, calculating sun exposure over time, and tagging wind-facing façades, we can guide orientation, volume shaping, and zone planning using only geometry. These scripts don't require a detailed model—they provide passive design intelligence that can reshape performance outcomes before a single floor plan is drawn.

16 – Projection Logic & Façade Classification

When working with mass elements, we can go beyond basic orientation. By analyzing surface projection angles and slope data, automation scripts help classify facades for daylight potential, glare risks, and envelope strategies. These classifications inform decisions about shading, glazing percentages, and view corridors. With

no walls in place, this analysis drives precision planning early—especially valuable when designing towers or complex volumetric forms.



17 – Conceptual Volume Slicing for Functional Zoning

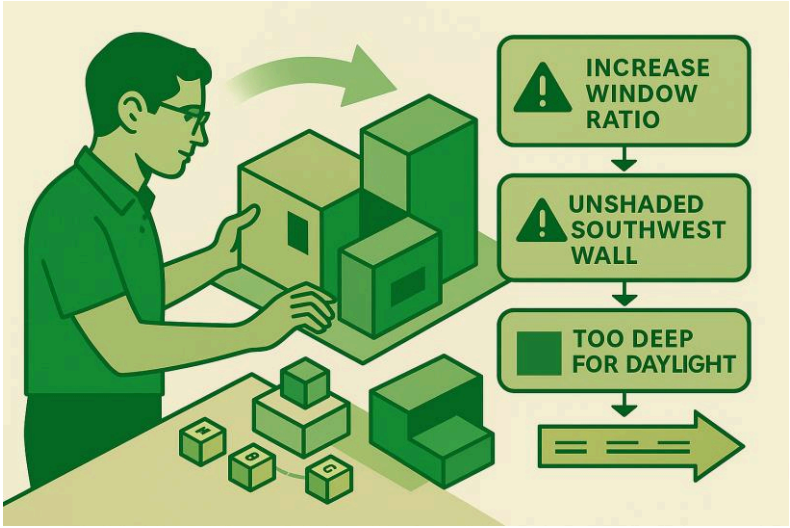
Not all energy logic happens on a flat floor. Automation tools allow us to slice mass forms into vertical and horizontal layers, using them to pre-assign volume zones by use type or environmental condition. Whether planning commercial podiums under residential blocks, or stacking thermal loads vertically, this script strategy helps prepare massing models for phased energy analysis and early stakeholder coordination—bridging schematic design with analytical thinking.

18 – Summary: Conceptual Stage Automation

Conceptual automation proves that energy thinking isn't something that starts after design—it can start before. These scripts bring geometric logic, climate awareness, and passive potential into early ideation. They reduce guesswork and encourage intentional design moves before detail clutters intent. Automating at the massing stage means making orientation, zoning, and solar logic visible—and it sets the tone for every downstream energy conversation.

19 – How Forma and Revit Talk: Linking Design Environments

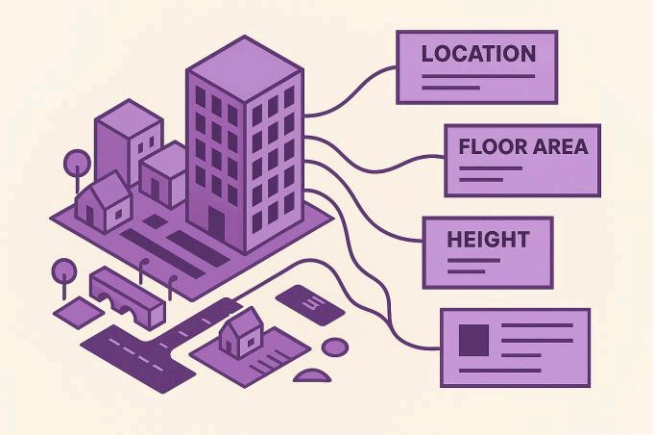
Forma isn't a separate silo—it's a conceptual design layer that connects seamlessly to Revit, giving designers early access to data that used to come too late. Through dedicated scripts and integrations, massing models and terrain data from Forma can feed directly into Dynamo for orientation analysis, zoning, and environmental simulation. Whether pulling solar exposure values, building height logic, or mass face directions, this connection allows Dynamo to interpret Forma data as early geometry with embedded performance potential. It's not about importing geometry—it's about integrating intelligence.



AIA25

20 – What Forma Brings to Early Performance Design

Forma enables decisions that matter, right where they begin: in early site studies, zoning envelopes, and schematic massing. It brings contextual clarity—sun paths, shadow studies, climate zones—and wraps them in a real-time environment where teams can test assumptions before modeling a single wall. Combined with Dynamo automation, Forma empowers architects to simulate solar access, calculate buildable volume, and optimize site layout, all while sketching. The real power isn't just in the data—it's in making that data visible, editable, and transferable to the tools that follow. Forma gives context a voice—and Dynamo lets that voice shape your design.

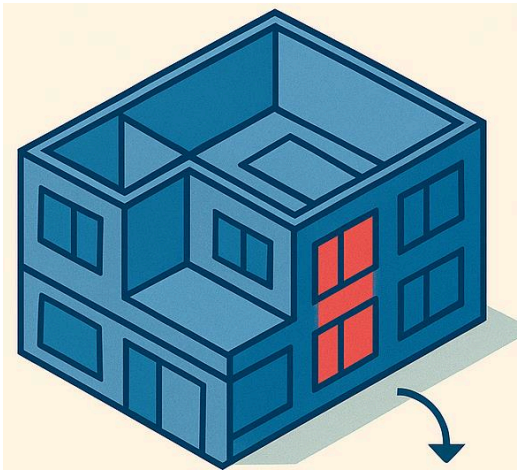


21 – Building Elements: Material Validation and Data Readiness

As design progresses and real walls, windows, and roofs enter the model, we shift from geometry to simulation readiness. These scripts ensure every element has the material data needed for meaningful energy modeling. They flag missing materials, check for thermal properties like conductivity, and prevent blind spots in the envelope. Before simulating, we must verify that Revit knows what the building is made of—and these tools automate that check.

22 – Glazing Logic and Shading Mapping

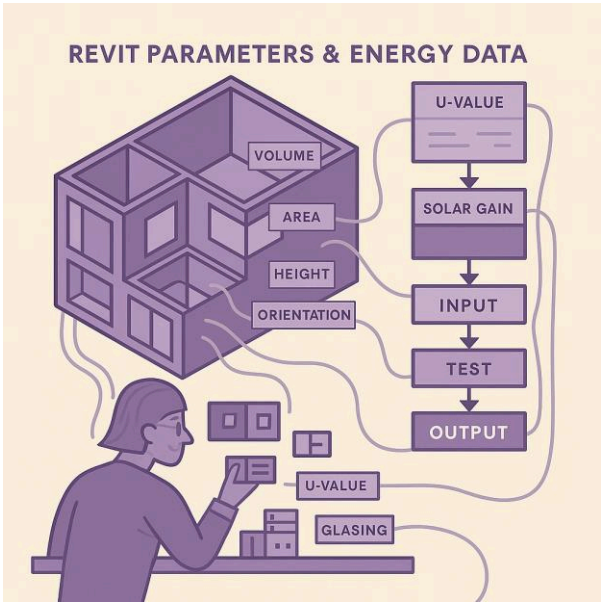
Once real elements exist, wall-to-window ratio becomes measurable—and automatable. With scripts calculating WWR per facade, comparing results to design goals, and visually flagging issues, this step translates code compliance into model clarity. Add shading logic from massing, and the building envelope becomes a feedback surface. These tools guide high-performance facade design by automating checks that would take hours to do manually.



AIA25

22b – Room Boundaries and Export Readiness

Energy analysis fails when boundaries are broken. Scripts in this stage ensure every wall, floor, ceiling, and window is correctly set to bound rooms. Once validated, models are prepared for export: to gbXML for simulation tools like Insight or OpenStudio, and to formats like IFC or NWC for team coordination. These automations bridge the gap between model authorship and model utility—ensuring what's drawn is ready to simulate or coordinate.



22c – Summary: Building Element Automation

Automating at the building element stage moves us from passive geometry to active performance. Material logic, envelope checks, and simulation prep are no longer left to chance or manual review. These scripts reduce simulation failure, improve export reliability, and accelerate compliance checks. If conceptual automation asks the question, "what could this shape become?"—this stage answers, "is this system ready to perform?"

AIA25

23 – Room & Space QA: Finding Errors Before Simulation

Rooms and spaces are often treated as placeholders—but in energy automation, they're simulation engines. Scripts at this stage catch what the model misses: elements that don't belong to rooms, roofs not bounding zones, rooms not placed or enclosed. These issues silently break simulations. With automated detection and reporting, they become fixable with clarity—saving hours of review and preserving trust in energy results.

24 – Space Logic: From Room to Energy Zone

Once the spatial structure is clean, automation turns architectural rooms into energy-ready spaces. Scripts create analytical spaces from rooms, pull parameters like internal gains or system type, and make this data visible for review. This is the leap from drafting to simulation—from modeling volume to modeling behavior. Every zone, every load, every EUI estimate depends on these scripts delivering clean, accurate space logic.

25 – Visual Overlays and Compliance Checks

Data without visibility is data ignored. These scripts apply color overlays and tag outputs for properties like EUI, gain intensity, or under-sizing. They show you where your rooms are too small, where your gains exceed threshold, or where compliance is at risk. With Dynamo doing the analysis and applying visual logic, the model becomes a dashboard—helping teams see problems in context, not just in spreadsheets.

26 – Summary: Room and Space Automation

Room and space automation completes the loop—where geometry, material, and logic converge. These scripts ensure models aren't just drawn right, but simulated right. They bring feedback into view, eliminate critical errors, and create a continuous path from design to analysis. In short: they stop simulation from being an afterthought, and instead make it a natural extension of your design process.

AIA25

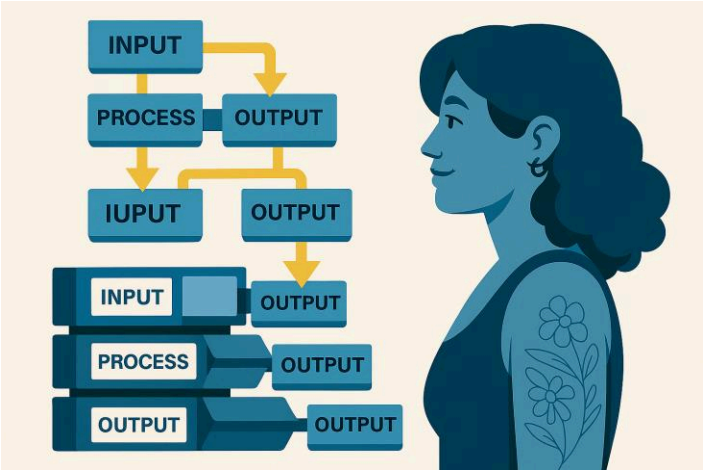
27 – Full System Summary: Script = Structure + Feedback

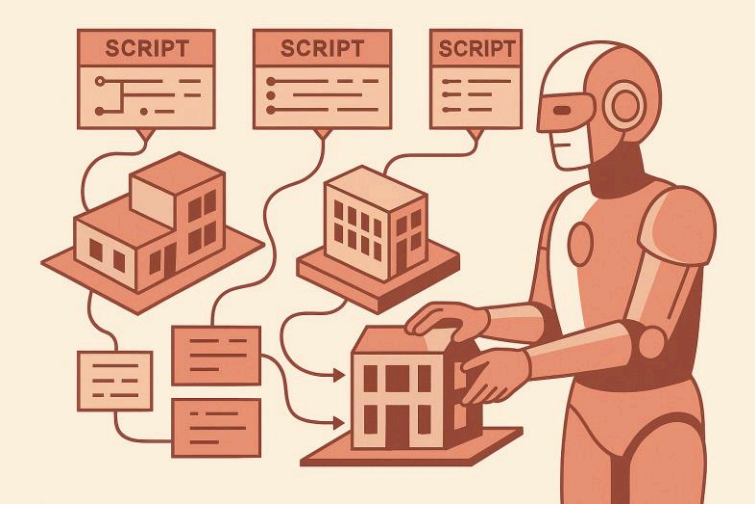
This presentation introduced not just scripts—but structure. From early orientation logic to space-based overlays, every phase was matched with purpose-built tools. The goal isn't to automate everything—it's to automate what matters, when it matters. Whether you're in massing, detailing, or documentation, energy performance isn't a separate step. It's embedded. Reusable. Visible. And with automation, it's finally scalable.

Get real-time energy feedback inside your design tools

28 – Logic and Automation

The future of design isn't about running complex simulations in the background—it's about embedding smart responses into the tools you already use. With Dynamo Player, you can launch logic-driven scripts that instantly check, tag, and export what matters most, without opening a graph. Whether you're validating floor area or pushing room data to a dashboard, you now have a real-time system that works with you—not against your timeline.





29 – Transitioning to Dynamo Player

Dynamo Player turns complex graphs into easy-to-use tools. With just a click, any team member—regardless of scripting background—can run a preconfigured script with sliders, dropdowns, and file pickers. Script 501 introduces this environment, showing how modular graphs built on best practices become accessible through a clean UI. This isn't just about accessibility—it's about workflow equity. Everyone contributes. No one waits on a "BIM expert" to push the button.

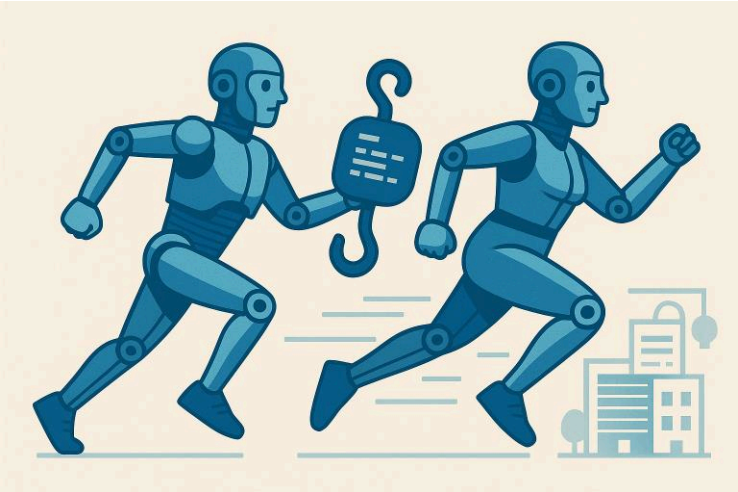
30 – Import External Data Seamlessly

Many early decisions live outside your model: site zoning, program requirements, occupancy data. We can use JSON or Excel files to pull in this information and connect it to Dynamo. You can import massing metrics, area

targets, or even terrain parameters—turning abstract values into inputs the model can understand. This bridges the gap between conceptual thinking and model logic, without manual typing or duplicate effort.

31 – Automate Exports with Multiplayer

Automate exports at scale. Using Bird Tools and structured logic, you can send IFCs, 2D views, and even annotated images from your model—without opening a dozen views manually. It's built for batch work: standardized outputs, predefined folders, and naming structures that make coordination easier across teams. Once set up, it becomes a one-click deliverable system—perfect for coordination rounds or consultant updates.



32 – Trigger Live Webhooks with Results

Why wait for emails, reports, or shared folders when your model can speak directly to the web? Dynamo can have triggers to webhook whenever a certain condition is met—like a room being over capacity, or a façade exceeding its WWR threshold. These triggers can push updates to Google Sheets, Notion, Airtable, Power BI, or Slack. Suddenly, your model isn't just a file—it's part of your digital coordination network.

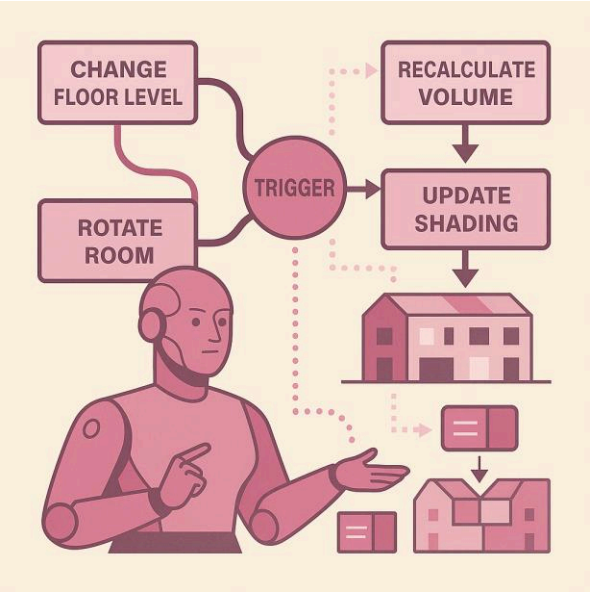
33 – Send Feedback Reports Where They're Needed

How about collecting QA or performance issues—and packing them. Whether it's a CSV of broken rooms, an image of the floor plan heatmap, or a JSON object of space data, this script sends it to the platform of your choice. QA isn't an isolated task anymore—it's something shared with everyone who needs to know. Architects. Engineers. Owners. All in sync.

34 – Visual Feedback That Lives in the Model

Forget toggling between Revit and dashboards. With the right scripts, you can apply visual feedback directly in the model: color-coded walls, tagged rooms, flagged errors—all live in the view. No exporting needed. This keeps your design logic visible, your coordination grounded, and your performance indicators where they belong: inside the model, not hidden in a spreadsheet.

AIA25



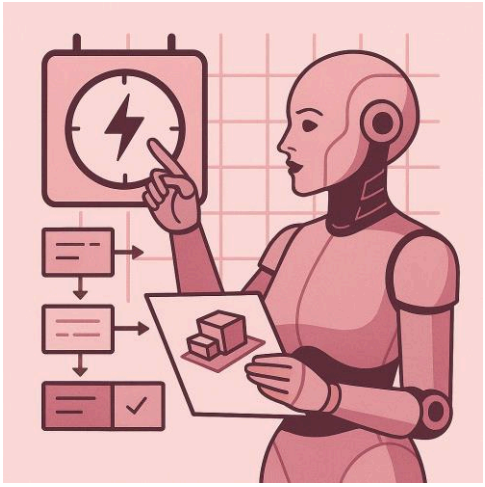
35 – From Logic to Triggers to Alerts

What makes real-time feedback powerful isn't just automation—it's **conditional automation**. The moment a room grows beyond its limit, the tag changes color. When the façade shifts orientation, the shading script re-runs. When something breaks your rule, you don't have to look for it—it shows up. This is design with awareness. Logic with memory. Coordination with speed.

36 – Scripted Tags = Design Intelligence

Tags don't just label. They inform. With scripting, a wall tag can say "70% glazing (limit exceeded)" or a room tag might show "Daylight OK" or "Needs Shading." These aren't just annotations—they're compliance checks, feedback devices, and coordination flags. And because they're logic-based, they update when the model changes—no manual touchups required.

37 – QA Isn't a Phase, It's a Loop



Most teams treat QA like a final check. But with scripting, it becomes a loop that runs every time something changes. Every save. Every sync. QA scripts catch missing tags, floating rooms, unbounded volumes, misassigned zones—and they flag them before it's too late. This means less rework, fewer design surprises, and more confidence in the energy outcomes your model is expected to deliver.

38 – Scaling Scripts Through Reuse

The value of automation multiplies with reuse. One well-structured script—like a WWR checker—can evolve into ten. Add a new threshold, swap the element category, or change the parameter output, and you've got a new tool. Teams that build structured logic build shared libraries. And teams that share, scale. You're no longer just modeling—you're evolving a framework.

AIA25

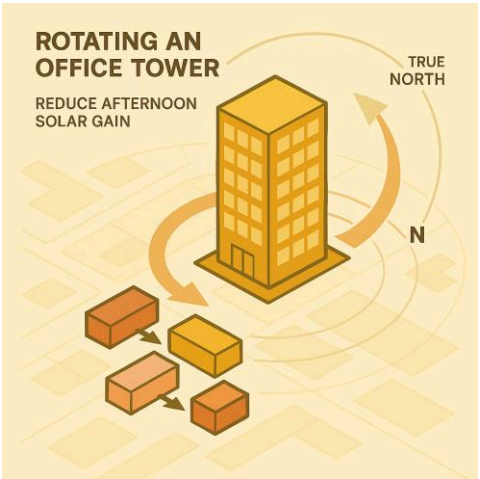
39 – You're Now Running a System, Not Just Scripts

This isn't about automating one task—it's about building a responsive system. You've connected inputs, parameters, geometry, and logic into a network that reacts, checks, visualizes, and exports—all without you manually managing every step. That's the shift: from isolated graphs to a workflow that moves with your design. Your model now speaks back. Your performance logic is no longer separate—it's embedded. The system doesn't just assist—it collaborates.

40 – This Is Energy Modeling on Autopilot

Welcome to the moment where energy modeling becomes part of the design flow—not a phase you survive later. You've seen how scripts extract data, overlay results, trigger alerts, and share insights—all from inside the model. That's the core of Energy Modeling on Autopilot: early insights, continuous feedback, and automation that keeps your design decisions informed from start to finish. You're not building tools anymore—you're building momentum. One graph. One trigger. One system. Fully embedded in how you work.

Target key design areas to improve performance and reduce costs



41 – Area 1: Early Massing Orientation & Solar Read (Logic)

Before walls exist, form orientation already determines energy impact. With Dynamo or Forma, you can automate solar vector alignment and surface exposure analysis. This lets your team assess and optimize mass orientation within seconds—guiding passive design with no modeling effort. It's not about simulating—it's about pointing the form in the right direction from day one.

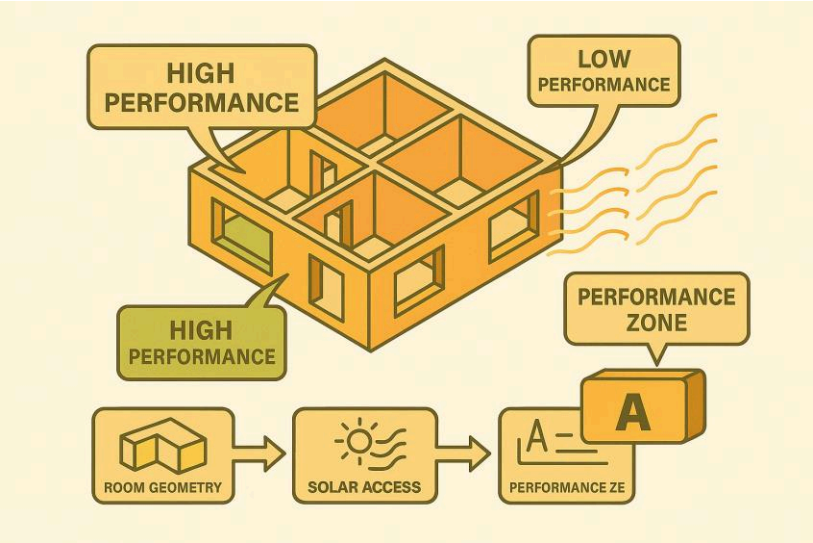
42 – Area 1: Early Massing Orientation & Solar Read (Benefit)

Performance Gain: Improves daylight access, reduces late-stage shading fixes, and optimizes passive gains.

Cost Benefit: Avoids costly facade redesign and helps meet climate zone strategies from the start.

🟢 *Savings: up to 5–8% of annual energy cost in hot climates.*

AIA25

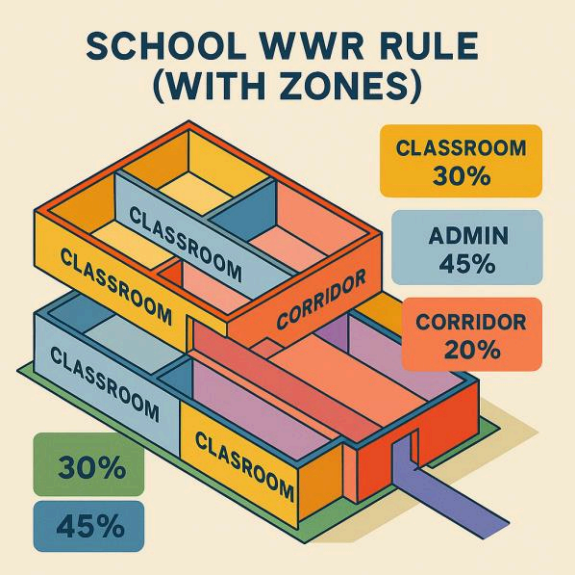


43 – Area 2: Material Validation Before Simulation

Energy models are only as accurate as their materials. With Dynamo scripts, you can scan every wall, roof, and floor for missing material assignments or thermal property gaps. No manual checks. No failed exports. Just instant, model-wide validation that confirms your envelope is simulation-ready.

Performance Gain: Enables correct U-values, surface conductivity, and simulation pass-through.

🟢 *Saves up to 10 hours per iteration cycle.*



44 – Area 3: Wall-to-Window Ratio Targeting

Instead of measuring by hand, use Dynamo to automatically calculate and flag WWR values per wall or facade. Override colors by threshold, add tags for clarity, and give your team an instant snapshot of what meets code—and what doesn't. No manual elevations. No schedule digging.

Performance Gain: Balances glazing, daylight, and insulation performance for better facades.

🟢 *Saves $8–12/m² in redesign costs.*

AIA25

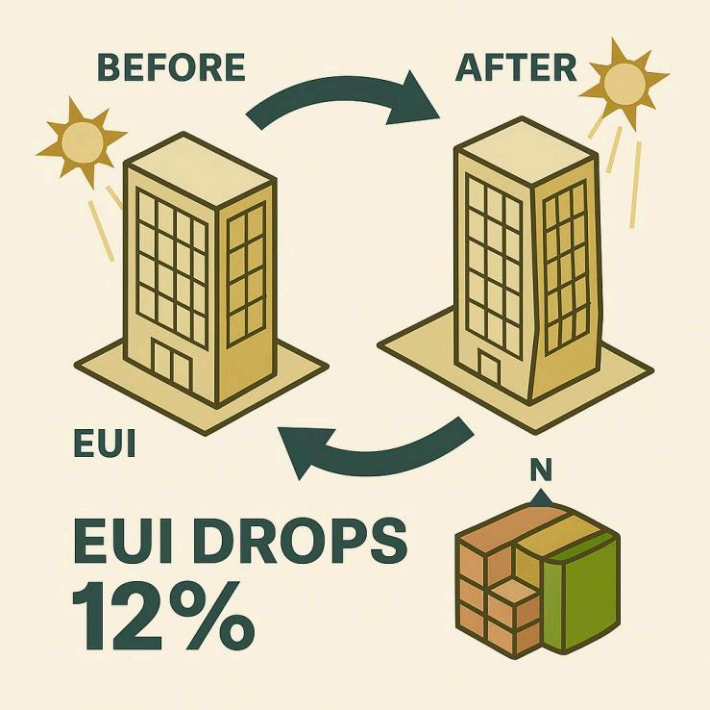
45 – Area 4: QA of Room and Space Integrity (Logic)

Many simulation failures happen not because of poor design—but because of poor boundaries. With Dynamo, you can catch unplaced rooms, open loops, or non-room-bounding elements automatically. These QA scripts make sure the model is simulation-ready before anyone presses "export."

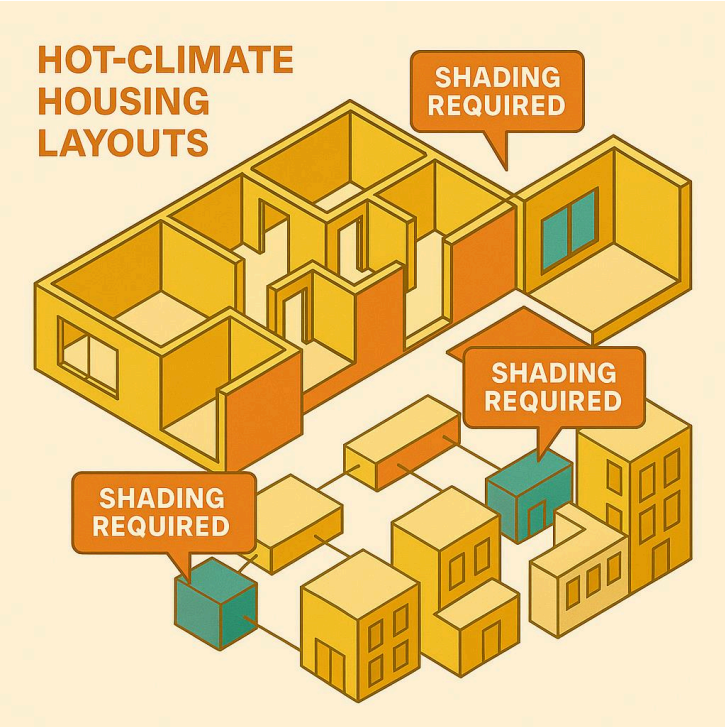
46 – Area 4: QA of Room and Space Integrity (Benefit)

Performance Gain: Valid zones ensure correct loads, volumes, and occupancy distribution.
Cost Benefit: Eliminates manual audit hours and improves early-stage coordination.

🟢 *Avoids $2k–$5k in rework from simulation breakdowns.*



AIA25



47 – Area 5: Undersized Room & Occupancy Analysis

Dynamo scripts can compare real room sizes to functional code minimums and flag anything undersized—before you submit. You can also auto-generate occupancy schedules per room type, keeping your load assumptions aligned with real space function.

Performance Gain: Prevents over/under-sizing HVAC systems and supports code-aware design.

🟢 *Saves 3–7% on HVAC installation and prevents redesign loops.*

48 – Area 6: Export Readiness & Documentation Sync (Logic)

Instead of guessing when your model is ready, trigger export logic only when all QA checks pass. Dynamo can package the gbXML, IFC, or QA report based on rule logic—no broken exports, no missing zones. Just a clean pipeline that connects model health to deliverables.

49 – Area 6: Export Readiness & Documentation Sync (Benefit)

Performance Gain: Enables smooth transition to simulation or consultant delivery.

Cost Benefit: Avoids delays in LEED, client reviews, and code checks.

🟢 *Speeds coordination and approval cycles by 15–20%.*

AIA25

50 – Recap Grid: Logic → Output → Benefit

Logic Check	Output	Result
Room Depth vs Code	Warning Tag	Daylight compliance
WWR by Room Type	Color Fill	Faster facade coordination
Material Missing	Alert Parameter	Valid gbXML export
Room Unplaced	QA Sheet Flag	Prevents zone calculation failure

This summary grid shows how each automation closes the loop: model input → logic → result → ROI.

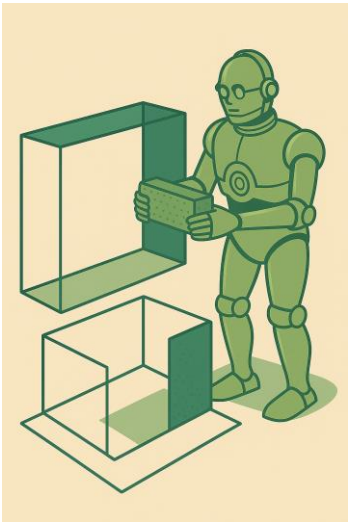
51 – Start With One, Build the Loop

You don't need a full automation system to get started. Pick one script—maybe WWR tagging, room QA, or solar mass rotation—and embed it in your next design. That single piece of logic will change how your team sees the model. From there, you scale. From one rule to ten. From reaction to prediction. From data chaos to automation clarity.

Use proven scripting practices to meet energy goals faster

52. Start Small, Repeat Often

The biggest mistake people make is trying to automate everything at once. Don't. Start with a single rule—like WWR or area checks—and make it solid. Then reuse that pattern. Once one rule is automated, it becomes a habit. From there, build a playbook of logic. You're not building a system in one shot—you're layering wins that compound. Repeatable structure beats large bursts every time.



53 – Modular Logic Design

"Every script in this system does one thing well—and all of them stack."
Great automation doesn't come from complexity—it comes from clarity. Modular scripts separate actions into clean parts: 🔵 *Input*, 🟠 *Logic*, 🟢 *Output*, 🔴 *QA*. This structure makes scripts easier to debug, easier to teach, and much easier to scale. Whether you're checking WWR or daylight logic, one block does one job. Build them like LEGO—not like spaghetti.
Benefit: Easy to debug, reuse, and scale across projects and teams.

AIA25

54 – Simulate Only When Ready

"Don't rush to Insight or gbXML—run pre-checks first."
Before you export or simulate, make sure your model is ready. This includes checking room boundaries, thermal parameters, and material presence. Use QA triggers to ensure simulation quality before it breaks. Clean logic avoids false results. With automation, this takes seconds—not hours.
Benefit: Prevents garbage-in, garbage-out simulations.

55 – Loop in Feedback, Visually

"Results don't belong in spreadsheets—show them in the model."
If feedback lives in Excel, it's forgotten. Use scripts to apply color fills, update tags, or overlay results in Revit views. Combine with webhooks to push images or summaries to dashboards. This creates awareness that's visual, not buried—and helps the team act without extra steps.
Benefit: Performance feedback embedded in daily modeling.

56 – Use Node Groups & Color Coding

"Let your logic explain itself visually."
Color conventions make logic instantly readable. Every script should use blue for input, orange for logic, green for output, and red for QA. This system creates continuity and makes complex scripts navigable. Anyone reviewing your graph will understand it in seconds—no training needed.
🧩 **Benefit**: Enables fast onboarding, clarity, and consistency.

AIA25

57 – Build & Use Custom Nodes

"Don't repeat yourself—abstract logic into reusable tools."
If you're copying a group of nodes more than once, make a custom node. With Zero Touch nodes, you can turn complex logic into one clean block. Package your most used checks, exporters, or color logic. This turns your Dynamo work into a library your whole team can use.

🧩 **Benefit**: Future-proofs automation; makes scripts scalable across teams.

58 – Adopt Clear Naming & Grouping

"Every script tells a story—give it structure."
Use clear prefixes for each group: Input_, Logic_, Output_, QA_. Group titles should describe their function and outcome. For example: QA_CheckRoomBoundaries, Output_TagHighWWR. This creates modularity and lets others use your scripts without guessing what anything does.

🧩 **Benefit**: Makes graphs self-documenting and maintainable.

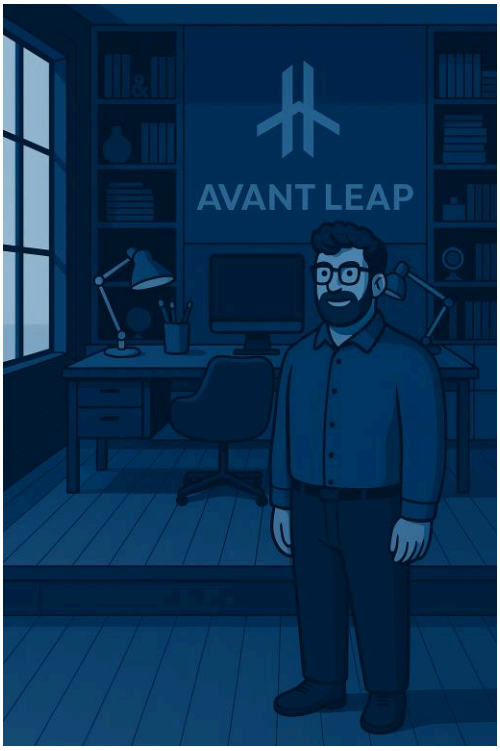
59 – Place Inputs at the Top, Outputs at the Bottom

"Graph structure should read like a page: top-down clarity."
Place inputs on the top left. Flow down and across to outputs and QA. Hide or collapse advanced logic unless editing. This helps new users understand your graphs quickly and keeps experienced ones working faster. The structure mirrors how people think—not just how Dynamo runs.

🧩 **Benefit**: Graphs become self-navigable and adaptable.

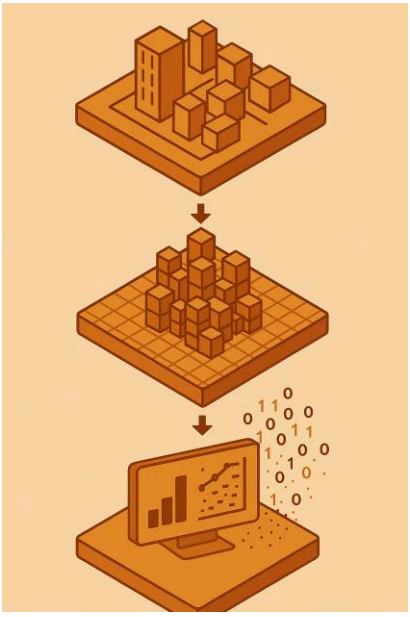
AIA25

AvantLeap's IP, Node Strategy & Development Tracker



60. Scripts Become Systems When You Reuse Logic

What starts as a one-off script grows into something much more powerful when you treat it as a reusable rule. Instead of "a script that checks WWR," it becomes "our company's rule for window design logic." You stop copying nodes—you start deploying patterns. That shift—from task to system—is the beginning of structured automation. This is where AvantLeap operates: turning repeated needs into shared infrastructure.



61. The 3 Zones Where Automation Lives

We've organized all our automation efforts into 3 working zones:

Zone 1: Early stage (Forma + Revit) where geometry is simple but logic is powerful.

Zone 2: Generative design with AI for spatial arrangement and logic-driven modeling.

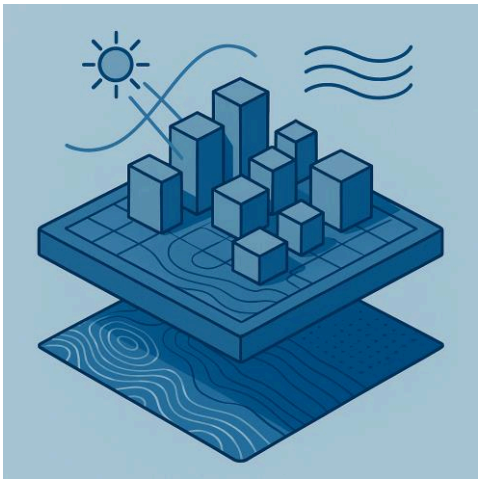
Zone 3: Digital twins and dashboards—where the design is done, but data still flows.

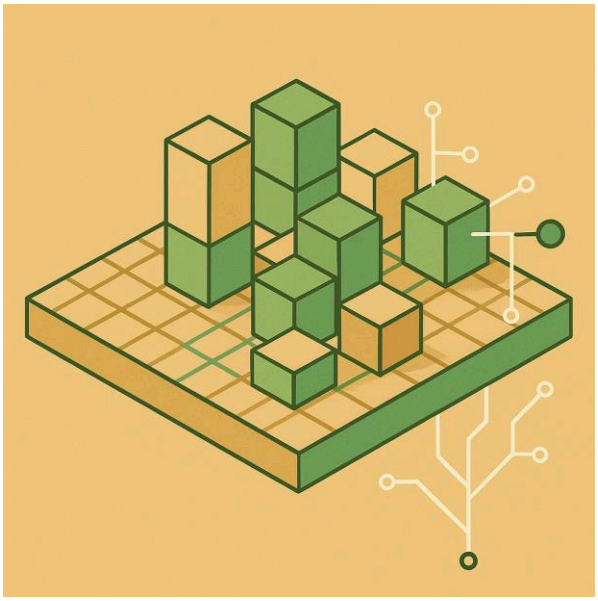
Each zone solves a different problem, but they all follow one principle: geometry + logic → visibility.

AIA25

62. Zone 1: Early Feedback with Forma + Revit

This is the most immediate win. You're working in massing or schematic design. Scripts flag glazing issues. Color zones by sun exposure. Suggest layout shifts before you've drawn a single detail. No BIM handover needed. This is where WWR checks, orientation triggers, and area-based zoning live. You're designing with feedback, not guessing then validating.





63. Zone 2: Generative Geometry + AI Logic

In Zone 2, we apply logic even *before* you place a wall. Imagine a wave-collapse model that fills rooms based on adjacency rules, daylight needs, or circulation paths. Or using AI to test multiple layout variations before committing to one. This zone isn't speculative—it's where VASA-style voxel analysis meets project rules. You're not automating design—you're co-designing with patterns.

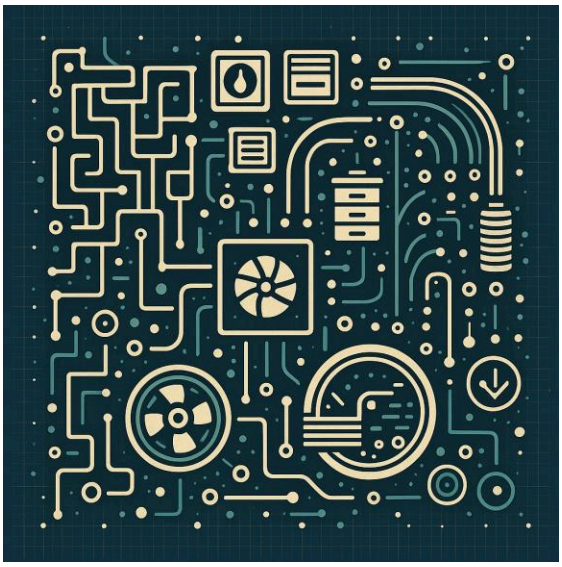
64. Zone 3: Digital Twin with Dashboard Intelligence

By the time the model's built, the geometry may freeze—but the data doesn't. Here, scripts don't alter the shape—they update parameters, synchronize IFC data, and populate dashboards. You're checking that assets are compliant. That room data is correct. That sensors and live updates feed into structured tags. We do this via APS, Airtable, or simple JSON flows—whatever makes sense for the client.

AIA25

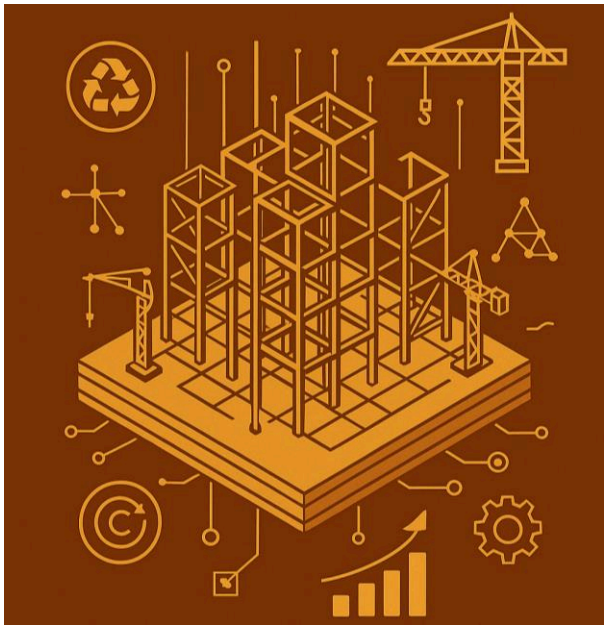
65 – AvantLeap's Node Strategy (6 Knowledge Areas)

AvantLeap organizes its Zero Touch Node development across six practical, high-impact domains. This structure ensures that each scripting tool aligns with real challenges teams face in modeling, coordination, automation, and sustainability. Instead of building generic nodes, we build *domain-specific accelerators*—designed for clarity, usability, and speed. Here's how each area translates into tangible workflows:

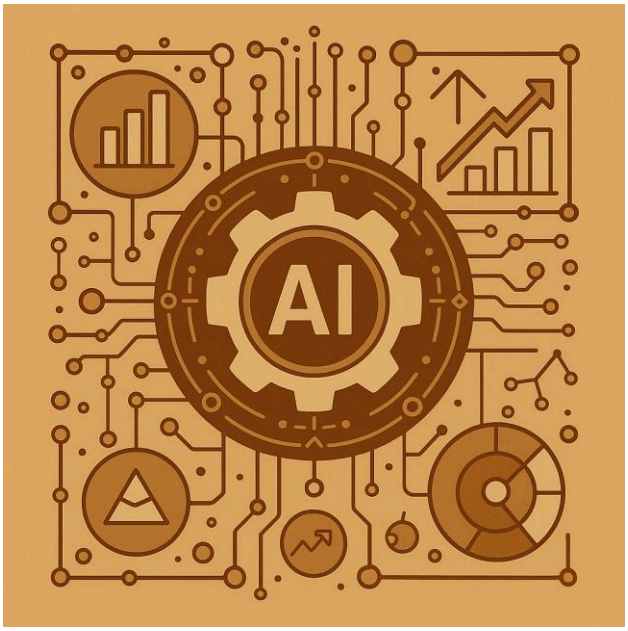


1. MEP (Mechanical, Electrical, Plumbing)
Our MEP nodes focus on accelerating routing decisions, auto-spacing of elements, clash-aware placement, and system validation. These tools help designers check clearances, minimize fittings, and script duct or pipe layout logic without needing full simulation tools. For example, a duct auto-routing node might minimize elbows while respecting system constraints, or a fire protection node might tag all sprinkler heads not within range of coverage. These are meant to reduce back-and-forth with engineers and empower early-stage layout refinement.

2. Structure Prefabrication
Structural nodes support component standardization, prefabrication planning, and connection logic. This includes tools that analyze repeatability of elements, tag beams or walls for prefab viability, or generate structural assemblies based on span and load constraints. These nodes simplify coordination between architecture and structural engineers and help spot issues like inconsistent spacing or over-spanned members early, leading to fewer RFIs and smoother fabrication planning.



AIA25



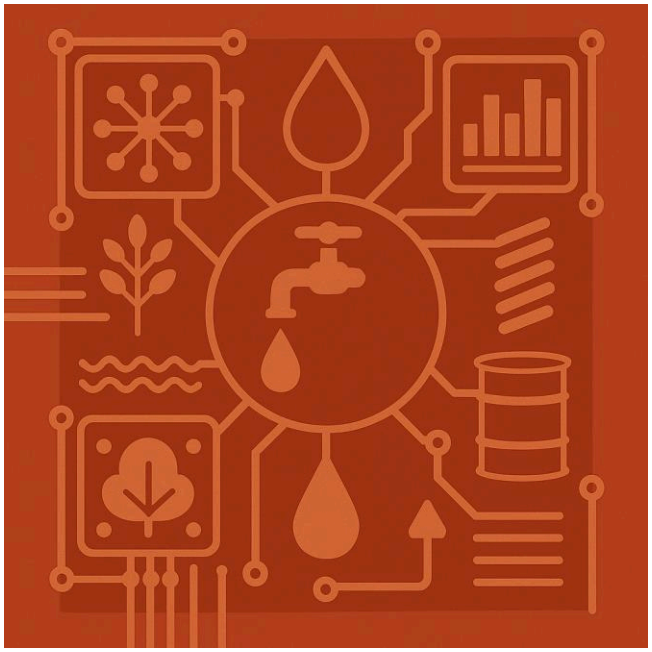
3. BIM Data Management

In this area, nodes are built to read, write, audit, and version-control BIM data across disciplines. These tools handle parameter QA, automated naming, data mapping, and synchronization tasks. Whether it's checking if a parameter is missing, syncing shared parameters between families, or creating a clean export for Power BI dashboards—these nodes turn the data buried in the model into something useful, clean, and report-ready. They also support data standards and reduce manual cleanup tasks.

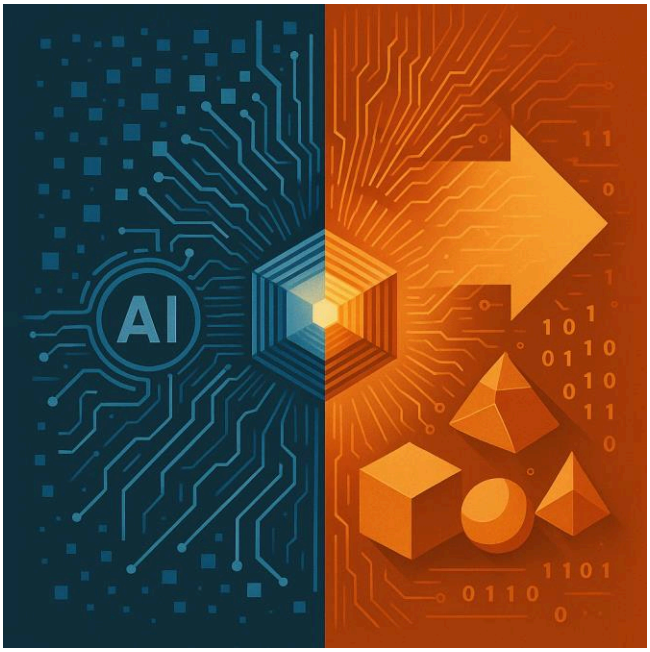
---

4. Energy & Water Logic

These nodes focus on performance-aware design logic. From scripts that track WWR and EUI, to zoning based on daylight or solar exposure, every node here helps the model provide *design-time feedback* before a simulation is needed. Some nodes connect to APIs like Forma or weather datasets to inform decisions, while others embed rules like "flag south-facing glazing over 40%." It's about performance logic made visible—without adding steps or exports.



AIA25

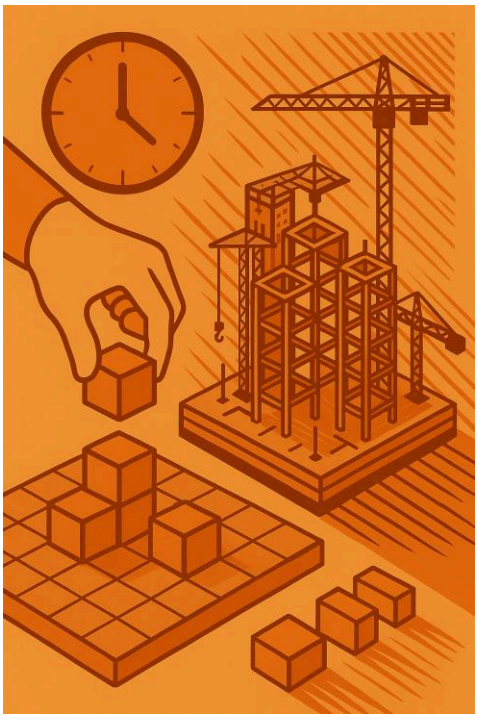


5. Stable Diffusion to 3D
This experimental category connects AI-generated images (e.g., via Stable Diffusion) to geometry generation in Revit or Dynamo. A façade pattern generated by AI becomes a curtain wall logic. A floor plan sketch becomes a series of layout options. These nodes bridge generative AI and BIM, letting designers prototype faster with pattern recognition, style-to-shape logic, or embedded AI suggestions—always traceable, always editable.

AIA25

6. Material Sustainability

Nodes in this group deal with material choice, embodied carbon, and assembly-level logic. They help flag materials that exceed carbon thresholds, calculate composite material impacts, or suggest alternatives based on project goals. These tools help teams answer questions like: *"Is this wall over our carbon target?"* or *"Which floor types offer a better footprint?"* without needing to leave Revit. They connect sustainability logic to modeling choices in real-time.



66 – Build Fast, Build Smart

Every node starts with the question: What can someone prototype in under 5 minutes? That's our design rule. No hidden complexity. No overbuilt graphs. Just one function, cleanly exposed. This lets teams test ideas fast, validate early, and stack up results without technical bottlenecks. It's not just about usability—it's about momentum. When logic is fast to write and faster to run, design exploration scales naturally.

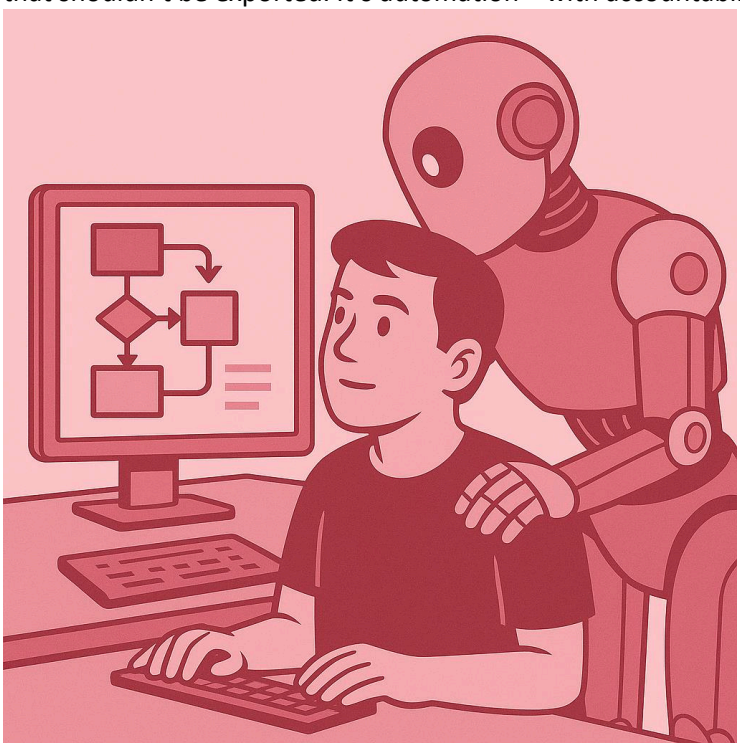
67 – Logic Comes Before Code

You won't see Python unless it's absolutely needed. Every node is designed first for logic: a decision, a rule, a threshold. If that logic works visually, we keep it that way. When complexity grows—branching, list operations, API calls—we migrate it to Python. This ensures that even non-coders can follow the structure, while advanced users can scale precision when needed. It's logic-first, not code-first.

68 – Security Is Part of the Architecture

AI can move fast—but without proper safeguards, it risks exposing data or logic. That's why our workflows are built with local/segmented data channels, and IP-protected node structures. Sensitive models are processed in secure environments, with AI endpoints either

obfuscated or self-hosted. Every automation respects the boundary of client data, and we don't expose logic that shouldn't be exported. It's automation—with accountability.



69– Agents and Audits

We use language-based models not just to generate ideas—but to audit logic. Paste a script description or export node outputs, and ask: "Does this logic match the goal?" AI doesn't need to build your Dynamo graph—it just needs to check if what's built makes sense. That's a peer review layer for every designer, and it helps us trust what we automate. It's fast, it's smart, and it's part of every QA loop we run.

70 – Lessons We've Learned

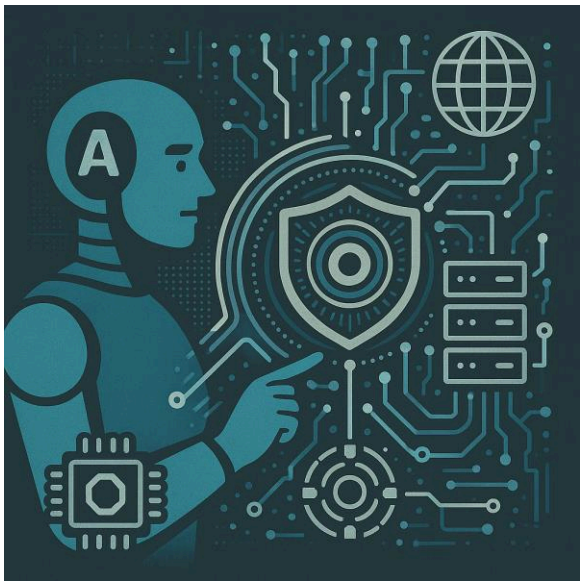
Title: What Building 100+ Scripts Has Taught Us
Paragraph:
Prototype Fast—Test logic with minimal inputs.
Keep Output Clear—Color beats spreadsheets.
Avoid Overreach—Don't simulate what you can check.
Build for Non-Coders—If your team can't use it, it's not helpful.
Expect Variation—Good logic handles mess, not just perfect models.

These five rules shape every script and node we share—because we're not building demos, we're building tools for real work.

AIA25

71 – Secure AI, Sustainable Outcomes

Every automation has a footprint—whether it's time saved, logic preserved, or risk avoided. But it also comes with responsibility. At AvantLeap, we embed sustainable logic (materials, energy, resilience) and secure logic (IP, client data, internal use) into every node. Automation can scale performance—but only if it respects the systems it supports. That's the model we follow: secure, sustainable, scalable.



Wrap-Up, Action, and Q&A

72. Tools, Logic, and Loops: Your Recap Grid

We'll leave you with a simple loop:

- **Design** in Forma or Revit
- **Script** your logic in Dynamo (WWR, room area, triggers)
- **Feedback** appears inside the model
- **Adjust** before it's too late

  It's not a platform. It's a rhythm. Once you know that flow, you can plug any logic into it.

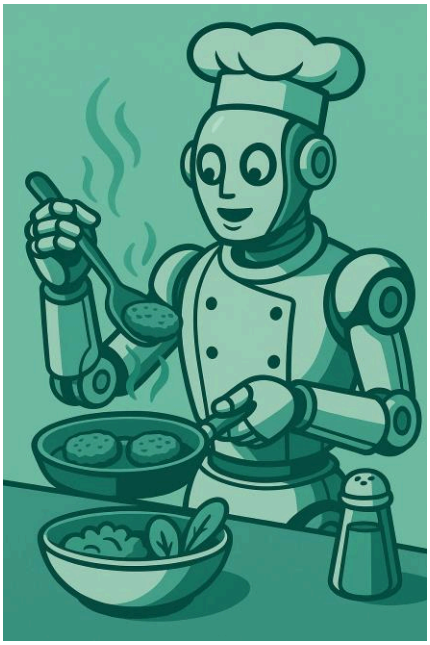
73. 10 Scripts in 10 Seconds

Here's a flash-round of wins you saw:

- Color rooms by performance zone
- Auto-check glazing percentage
- Alert on deep rooms for daylight
- Flag walls without orientation
- Tag rooms by EUI class
- Highlight missed parameters
- Apply climate-based shading rules

- Update dashboards from model data
- Trigger checks when floor level changes
- Generate summaries for documentation

These aren't future features. They're working today.



74. What You Can Actually Do on Monday

This isn't about becoming a Dynamo expert. It's about starting small:

- **Choose 1 rule** you apply by memory
- **Write it out as logic** (if/then)
- **Build a 5-node script** or use a template
- **Apply it** in a test model and **color the result**

Now you have performance *inside* your model. That's a win in under an hour.

75. How to Share This With Your Team

Don't go back and say, "I saw some advanced automation thing."

Instead say, "We can check WWR before code review."

Say, "We can color rooms by risk zones."

Say, "We don't need to guess when to shade anymore."

That's how adoption happens—through benefits, not terminology.

AIA25

76. From Chaos to Clarity (The Real Story)

Let's zoom out: your week is packed, decisions come fast, and performance usually waits until it's too late.
But what if feedback could keep up with your pace?
That's what this was really about:

- Making logic visible
- Making design decisions smarter
- Making your model work harder, *without* working more

77. Join the Conversation (Not the Platform)

We're not asking you to buy software or change ecosystems. We're inviting you to try one script. See what it unlocks. Share your logic. Tell us where it breaks. Download the handout. Grab a template. Fork the GitHub. You don't need to wait for a rollout. This starts now.

78. Q&A + Final Logic Drop

Let's open the floor. Questions, ideas, challenges—let's talk. And as a final gift: we'll drop one bonus script in the handout—an auto-updater that syncs geometry changes to a live Airtable dashboard. Small, fast, and useful. Just like the whole idea here.

AIA25

Dynamo simples

101: Geometry Extractions
Goal: Understand how Dynamo pulls model geometry
Steps:
• Get all walls, rooms, and windows from the model
• Display element count or type in Watch node
• Visualize element locations with geometry preview
Result: Familiarizes users with Dynamo's model-reading capabilities

---

102: Parameter Extractions
Goal: Learn how to access model data for automation
Steps:
• Get room elements
• Extract parameters like Area, Volume, Name
• Display values or write to output panel
Result: Builds foundation for metric-based logic

103: Intersect With Elements
Goal: Determine spatial relationships between objects
Steps:
• Get rooms and architectural elements (walls, ducts, etc.)
• Use geometry intersection nodes
• Identify overlap or adjacency by geometry logic
Result: Enables adjacency analysis for routing and performance modeling

---

104: Color Override for Elements
Goal: Connect logic output to Revit visualization
Steps:
• Get all exterior walls
• Evaluate WWR or orientation logic
• Apply color override if value exceeds threshold
Result: Logic becomes instantly visible in the Revit model

105: Set Parameter Values
Goal: Write calculated results back into the model
Steps:
• Calculate area-to-volume ratio
• Compare against standard threshold
• Write result into a custom parameter “Perf_Rating”
Result: Makes scripts part of the model’s data history

AIA25

201: Assign Solar Orientation to Mass Faces
Goal: Tag each facade face with its orientation
Steps:
• Get conceptual mass surfaces
• Calculate face normal and azimuth
• Set parameter or text label based on direction
Result: Helps drive passive design logic early on

202: Mass Face Projection Checker
Goal: Visualize projection angles for solar performance
Steps:
• Collect mass surfaces
• Calculate angle from vertical or horizontal
• Output color-coded results or tag projection angle
Result: Identifies surfaces with high solar exposure risk

203: Sun Path Visibility Overlay
Goal: Show which mass faces receive sun vs. shade
Steps:
• Input date/time
• Calculate sun vector
• Compare to face normal → tag or color based on angle
Result: Highlights surfaces needing shading before walls exist

---

204: Wind Direction Exposure Logic
Goal: Tag facades most exposed to prevailing winds
Steps:
• Define wind direction based on climate zone
• Analyze face normals vs. wind azimuth
• Tag faces or apply visual override
Result: Supports early facade planning for wind mitigation

205: Conceptual Volume Zone Slicer
Goal: Cut mass into stacked slices for early zoning
Steps:
• Get massing volume
• Slice by floor height, zone use, or height band
• Tag or isolate each slice
Result: Supports schematic allocation of use types or zones

301: Validate Material Presence in Elements
Goal: Ensure all geometry has assigned materials
Steps:

- Get all walls, floors, roofs, windows, doors
- Check for null material assignment
- Color or tag if material is missing

Result: Prevents simulation errors caused by empty material slots

---

302: Check for Thermal Properties in Materials
Goal: Validate simulation-readiness of materials
Steps:
- Access materials from elements
- Check for required thermal properties (U-value, conductivity)
- Flag missing or placeholder values

Result: Ensures meaningful energy results from simulation tools

---

303: Wall-to-Window Ratio
Goal: Automatically calculate and flag WWR violations
Steps:
- Collect walls and their hosted windows
- Compute WWR = glazing area ÷ wall area
- Tag walls with WWR value; color if above threshold

Result: Detects facade risks visually before code checks fail

---

304: Map Mass Faces to Wall Elements for Shading
Goal: Connect conceptual shading intent to real model elements
Steps:
- Identify mass faces with shading logic
- Map those faces to real wall locations
- Apply a linked shading status or tag

Result: Brings early passive logic into developed geometry

305: Classify Exterior vs Interior Walls
Goal: Identify and correct wall type misclassifications
Steps:
- Analyze wall position relative to bounding zones
- Check Revit's "Function" parameter
- Override color if mismatched

Result: Ensures boundary definitions are simulation-compliant

---

306: Check Room-Bounding Settings on Elements
Goal: Ensure all bounding elements contribute to rooms
Steps:
- Get walls, floors, ceilings, roofs
- Check if Room Bounding = True

• Flag those set incorrectly
Result: Prevents unbounded zones and volume calculation errors

---

307: Export Energy Model to gbXML
Goal: Confirm model readiness for gbXML export
Steps:
• Run QA checks on geometry, parameters, bounding elements
• Trigger export process via Revit API or UI hook
• Log results and verify file output
Result: Streamlines the path from design model to energy simulation

---

308: Export Coordination Models (IFC, NWC)
Goal: Package coordination models from valid energy geometry
Steps:
• Get valid views and geometry
• Export NWC or IFC with proper naming and filters
• Archive or share output
Result: Facilitates external collaboration from energy-ready files

---

401: Detect Elements Outside of Any Room
Goal: Identify modeled elements not enclosed by rooms
Steps:
• Compare element location to room boundaries
• Flag elements with null Room assignment
• Tag or color for visibility
Result: Improves modeling integrity and simulation enclosure

402: Roof-Room Intersect Validator
Goal: Ensure rooms intersect with roof geometry above
Steps:
• Check room bounding box top against roof planes
• Identify rooms with no enclosing roof
• Output result in view or tag
Result: Supports volume and envelope completeness

403: Unenclosed Room Detector
Goal: Catch rooms missing full enclosure
Steps:
• Analyze room boundary segments
• Check for gaps, non-closed loops
• Output QA status
Result: Prevents simulation failures due to broken boundaries

---

AIA25

404: Flag Missing or Unplaced Rooms
Goal: Identify unplaced or invisible rooms
Steps:
• Get all rooms in project
• Filter out rooms with area = 0 or unplaced status
• Write “Needs Placement” into a QA parameter
Result: Prevents broken zones and invisible simulation gaps

405: Verify Room Naming Standards
Goal: Standardize and validate room naming
Steps:
• Check Name and Number parameters
• Compare to naming templates or expected values
• Flag duplicates or blanks
Result: Improves clarity, documentation, and schedule accuracy

---

406: Auto-Create Spaces from Existing Rooms
Goal: Build energy-ready spaces directly from rooms
Steps:
• Get all rooms
• Use Create Space by Room logic in Dynamo
• Assign matching parameters
Result: Simplifies space creation for energy simulation workflows

---

407: Extract and Review Space Properties
Goal: Pull and analyze simulation parameters from spaces
Steps:
• Get all spaces
• Extract gain, load, zone type, etc.
• Output as a report or tag
Result: Enables fast review of simulation data without exports

---

408: Detect Under-Sized Rooms by Function
Goal: Flag rooms that don’t meet minimum program sizes
Steps:
• Get all rooms
• Read Function and Area
• Compare against space-type threshold dictionary
• Apply warning tag or “Flag = Yes” if below threshold
Result: Avoids spatial layout issues and early code violations

---

409: Overlay Space Data Visually
Goal: Visualize simulation metrics in Revit views

AIA25

Steps:

- Get all spaces and a key metric (e.g., EUI, volume)
- Set a parameter or tag text
- Apply view filters to color based on value

Result: Makes performance data readable at a glance inside the model