

hackathons and case competitions: putting classroom theory to the test at aelo swiss academy

There is a particular kind of silence that falls over a room right after a case brief gets opened. Forty-some students, one shared PDF, and a countdown clock on the wall screen that has just started doing its job. Someone reads the first paragraph out loud. Someone else is already uncapping a marker. And somewhere in the back, a student who aced every written exam this semester quietly realizes that the framework from Tuesday's lecture does not tell her what to do first.

That moment is the whole reason we run hackathons and case competitions at AELO Swiss Academy. Not as decoration on the calendar, and not as a break from "real" teaching. They sit close to the center of how the place works, because the distance between knowing something and being able to use it under pressure is exactly where most learning either happens or quietly doesn't.

why a timer changes everything

In a classroom, problems arrive pre-cleaned. The question is stated, the relevant data is provided, the distractors have been politely removed, and there is usually a sense that a correct answer exists. That is fine for building foundations. It is also a slightly dishonest picture of how work actually feels.

In a case competition, the brief is messy on purpose. There is too much data in one place and not enough in another. Two goals quietly conflict with each other and nobody will point that out unless you notice. The clock never pauses while you debate which of three plausible interpretations of the client's question is the right one. Students have to make judgment calls with partial information, defend them in front of strangers, and live with the consequences of their choices for exactly as long as it takes a judge to raise an eyebrow.

Theory stops being a chapter you cite and becomes a tool you reach for. A Porter's five forces sketch, a break-even calculation, a quick user journey map. These things stop being homework artifacts and start being instruments. You can almost see the shift happen over a weekend. On Saturday morning students ask "what did we cover about this?" By Sunday afternoon they ask "what does this decision need from us right now?"

two formats, one idea

People sometimes use "hackathon" and "case competition" interchangeably. They overlap, but they reward different muscles, and we run both deliberately.

| | hackathon | case competition |

|---|---|---|

| Raw material | an open-ended build, usually digital | a written brief with data and a client question |

| Deliverable | a working prototype or demo | a recommendation, usually a short deck and a pitch |

| Typical window | 24 to 48 hours | anywhere from 24 hours to a week |



| Where teams lose | polishing features instead of shipping | researching forever instead of deciding |

A hackathon punishes perfectionism. If nothing runs by Sunday, nothing counts, no matter how elegant the architecture diagram looked. A case competition punishes indecision. You can have the most thorough analysis in the room and still lose to the team that made a clear call and defended it well. Same clock, different failure modes.

anatomy of a competition weekend

Most of our weekends follow a rhythm that has settled into place over the years, refined after each event by what students told us afterward.

The brief drops on Friday evening. Teams of four or five get the document, a dataset of varying quality, and a hard deadline for Sunday afternoon. Saturday is for structuring the problem, splitting the work, and hitting the first wall, which usually arrives mid-afternoon when the team realizes their initial framing doesn't survive contact with the numbers. Mentor rounds run every few hours: a mentor sits with each team for fifteen minutes or so, asks uncomfortable questions, and leaves without giving answers. Sunday morning is deck freeze, which is exactly what it sounds like. After that, no more edits, only rehearsal. The afternoon belongs to pitches, usually ten to fifteen minutes per team plus questions from a panel of three to five judges drawn from industry and faculty.

The most important hours are rarely the ones people photograph. They are the hours around midnight on Saturday, when a team has to cut half of what they built because they finally understand what the question was actually asking. Watching students negotiate that cut, badly at first and then better with practice, is more educational than most seminars I have sat through.

what the classroom cannot simulate

Three things only show up when the clock is running.

The first is ambiguity. Real briefs don't tell you which problem to solve, only symptoms and constraints. Choosing the problem is half the work, and there is no answer key to check against until the judges speak.

The second is teamwork under fatigue. Everyone is pleasant at Friday kickoff. By hour twenty, you learn who goes quiet under stress, who papers over disagreement with consensus, and whether the team can disagree about an idea without it becoming a disagreement about a person. These dynamics exist in group projects too, but a deadline measured in hours rather than weeks strips away the room to avoid them.

The third is the cost of a wrong abstraction. In an essay, a shaky assumption costs you a paragraph of feedback. In a pitch, it costs you the moment a judge asks a follow-up question and the whole recommendation wobbles. Students feel that in their stomach, which is precisely why it sticks.

three moments that taught more than a lecture

A few stories from past events keep getting retold, usually by the students who lived them.

One team at an early hackathon decided to build an app, a dashboard, and a pricing [reddit post](#) model in thirty-six hours. Ambition is not the problem there, arithmetic is. They shipped a prototype that crashed during the demo and spent their pitch apologizing. The lesson landed hard: in a fixed window, scope is the only variable you truly control. The next semester, a different team entered with the opposite instinct. Three slides, one well-sourced number, and a recommendation so clear that one of the judges repeated it from memory during the debrief. They won. Not because they worked less, but because they decided earlier what their one argument would be.

Another time, a dataset in a retail case had a seasonality quirk buried in the monthly figures. Most teams ran the numbers straight and recommended a pricing change that would have backfired every summer. One team noticed the pattern, reframed the entire problem around it, and asked the judges a question back: "assuming this pattern holds, is growth even the client's real problem?" That is the moment where analysis turns into judgment, and you cannot schedule it into a syllabus. You can only create the conditions where it might happen.



Then there is the quieter story, the one about the teammate who said almost nothing all weekend and rebuilt the entire financial model at one in the morning after spotting an error nobody else caught. Teams learn that contribution is not airtime. That lesson tends to arrive around hour thirty and never really leaves.

what judges actually reward

Judging varies by event, but after watching enough panels, patterns emerge. Judges are not looking for the most slides or the fanciest charts. They are looking for evidence of thinking. When we brief our panels, we point them [flight training](#) toward five things, and we share the same list with students beforehand.

1. A recommendation you can state in one sentence without hedging.
2. Evidence that actually connects to that recommendation, not a museum of everything the team found.
3. Honest acknowledgment of trade-offs and what could go wrong.

4. Feasibility within constraints a real client would recognize, including money, time, and people.

5. A question round where the team thinks out loud instead of retreating to a script.

Notice how little of that is about polish. A shaky delivery with a sound argument beats a smooth delivery with a hollow one more often than students expect. Judges forgive nerves. They do not forgive a recommendation that evaporates under the second question.

the honest downsides

It would be dishonest to present competitions as pure upside, and students deserve to hear the trade-offs from us rather than discover them at hour thirty-one.

Sleep is the obvious one. A weekend that runs past midnight twice in a row produces worse decisions on Sunday, and everyone knows it and does it anyway. We now build a suggested shutdown time into the schedule and remind teams that tired teams make expensive mistakes. Some ignore it. The ones who don't tend to pitch better, which becomes its own lesson.

There is also a subtle risk of learning to game the rubric rather than to think. Competitions reward legibility, and legibility is not the same as depth. A team can optimize for what scores well and walk away having practiced performance rather than analysis. We push back on that with mandatory debriefs in the following week, where we look past the winning decks and ask what each team would have done with two more weeks. The answers are usually more interesting than the pitches.

And there is an equity problem worth naming. Teams whose members already know each other start with trust that strangers have to build from scratch under pressure. We mix teams deliberately for exactly this reason, even though it makes the first half of every Saturday more awkward. The awkwardness is the point.

how we prepare students at aelo swiss academy

Nobody should walk into their first competition cold. At AELO Swiss Academy, the lead-up is deliberately unglamorous.

Practice rounds use briefs in the style of past events, run at half length, so students experience the whole arc before it counts. Roles rotate: whoever built the financial model last time presents this time, because the goal is range, not early specialization. Mentors are coached to ask questions rather than hand out answers, which frustrates students for about two practice rounds before it starts working. And every event ends with a written debrief, both from the judges and from the teams themselves, on what they would keep, cut, and change.

The keyword across all of it is reps. One competition is an experience. Three or four across a program of study is a habit of mind, and that habit shows up later in job interviews, client work, and the general ability to walk into a messy situation and find the question worth answering.

a short checklist for your first weekend

For students about to do this for the first time, a handful of things reliably separate good weekends from rough ones.

1. Read the brief three times, then write the client's question in your own words and get every teammate to agree on that sentence before anything else.
2. Fix your scope before you touch your tools. Decide what you will not do, in writing.

3. Put one person in charge of the numbers. Split ownership of data and you will present two different totals to a judge.
4. Build the deck around the recommendation, not around the research journey. Judges want the destination.
5. Sleep at least a few hours. Everything you do after hour thirty is worse than what you would do fresh on Sunday morning.

None of this is complicated. All of it is hard to remember at eleven at night with sticky notes covering every surface, which is why it is worth deciding in advance.

what stays with you afterwards

The prototype from a hackathon usually dies within a month, and that is fine. The deck from a case competition gets filed away. What lasts is stranger and more valuable: the reflex to ask what problem we are actually solving, the habit of stress-testing a recommendation before someone else does it for you, and the memory that you once made a defensible call with incomplete information and a clock running.

Alumni come back to judge, which says something. They sit where the panel sits, ask the same style of questions that once rattled them, and then tell current students the thing nobody believed at the time: the weekend was less about the case than about finding out how they think when it matters. Watching that loop close, semester after semester, is the quiet proof that putting theory under a timer works. Not every time, not for everyone in the same way, but reliably enough that we keep printing the briefs, opening the room, and starting the clock.

