How on earth do you put an ambition to use AI in education into practice?

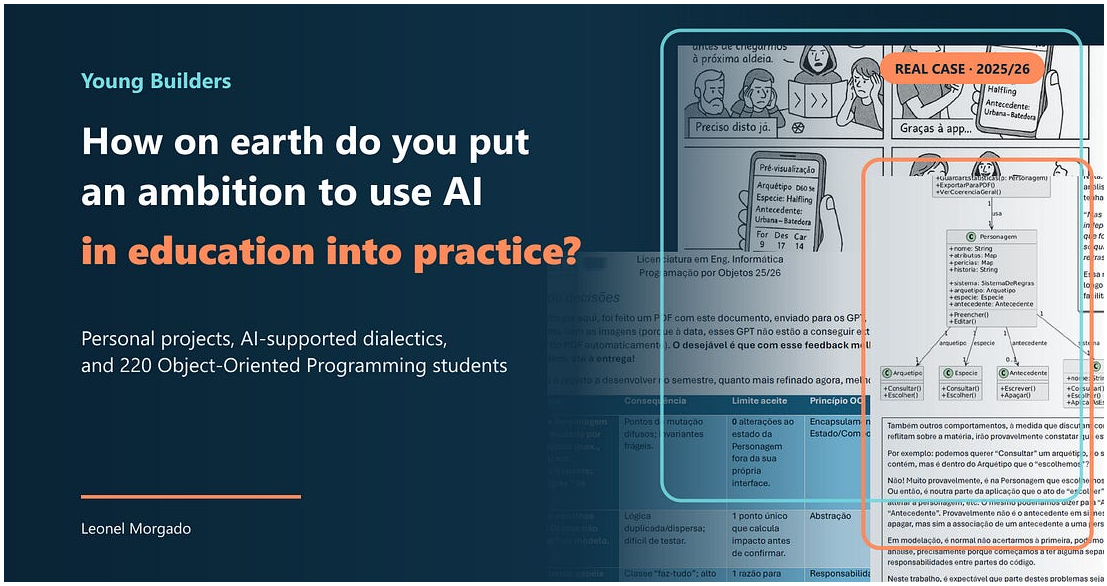
One case involving personal projects, AI-supported dialectics, and 220 Object-Oriented Programming students.

LEONEL MORGADO

JUL 11, 2026

🇵🇹 There's a Portuguese language version.

https://doi.org/10.5281/zenodo.21309211



Not long ago, I wrote here that AI will force us to take seriously educational ambitions we have been proclaiming since the nineteenth century: understanding, purpose, personal appropriation, the ability to judge, decide, and take responsibility. Those are fine ideas. Defending them in an essay is relatively easy.

Then Monday comes.

Those ideas have to appear in concrete activities, with assignment briefs students can interpret and work that fits the course credits. Students need support (in my case at Universidade Aberta, asynchronously and in writing, through Moodle forums). Everything has to fit the deadlines. And then it has to be assessed. All of this while students use GPTs or other AI systems. I strongly encourage them to do so, but they would still use them if I discouraged them. They may use them well, use them badly... use them critically to explore a difficulty, or simply use them to produce something that fills the allotted space in an assignment. An instructor's work should not consist of naively drawing boundaries around when the tools may be used, nor should it waste away trying to detect their tracks in pursuit of a chimera. At least I do not think that is a useful way to practise our profession. I am more interested in raising the ambition of what students are to learn and in finding processes that are workable for the general body of students and educators. In other words, processes that are not reserved for those who already have high motivation, commitment, resilience, and dedication. On the contrary, processes that work within the complex reality of human beings as they are, including those who would probably never get there without guidance, support, or enriching experiences.

This essay is about one of my concrete attempts. I carried it out during the 2025/26 academic year, in an Object-Oriented Programming course

on the Informatics Engineering degree at Universidade Aberta, with 220 students (give or take: the number does not remain stable over the semester). Do not expect me to present wondrous results showing learning gains. Finding out whether something in education is better than something else, or even better than doing nothing, is an unrewarding, time-consuming task, because we cannot take the same students through the same experience twice. Knowing that will take years and meticulous research into the plans, their execution, and so forth. What I have now is an activity design that was actually used, something real rather than merely imagined. I have the artefacts that gave it form, observations from its implementation, and a considerably clearer idea of what seems to have gone well and what I aspire to improve.

What remained untaught when we taught Object-Oriented Programming

If you come from another field, do not be put off by this being about computer science. I think it will be useful for (almost) everyone. A brief explanation will be enough to establish the context. Object-oriented programming is a way (let us call it that) of organising the structure of computer programs so that they remain easier to understand, modify, and manage as they grow and as many hands and minds start tampering with them, rather than just one skilful software craftsperson. It uses several techniques for this purpose, with names that reflect ideals, such as “encapsulation”, “overloading”, “inheritance”, and “polymorphism”. Their precise meanings do not matter for this essay. It is enough to understand that they are techniques for organising and structuring complex things.

In practice, pressures of scale (the number of students), heterogeneity (the variation in each student’s level of attainment), and time (do I really need to explain?) meant that much of the teaching ended up concentrating on the expression and practice of syntax and techniques.

There are good reasons for this, mind you. They are complex, they require repetition and experience, the code has to work, and the most basic errors may stop a student from moving forward. It is possible (common, in fact) to teach encapsulation without the student ever feeling what problem the technique has solved. A student may implement inheritance without developing any criterion for deciding when it will make a program easier to evolve and when it will instead trap the program inside a poor structural decision.

A customary solution is to develop individual projects. When someone tries to give form to an idea that is genuinely their own, the classes stop aping examples prepared by the instructor and begin to reflect responsibilities towards that idea. The devil emerges in the details: is the state of each element coherent? Do we allow something to change or be replicated, or impose it as static and unique? Do we identify common practices, collaboration, and trust, or do small differences generate boundaries and separation? (Easy, computer-science colleagues, I am writing it this way so that people outside computing can follow the general argument.) It is in these details, piece by piece and line by line, that understanding the principles proves useful.

This path of project-based learning has (had?) a logistical problem. Regularly following the projects of more than two hundred students, especially at an online university, would inevitably distribute feedback unevenly (even with tutors supporting groups). Some students would ask questions early, when teaching pressure was lower, and might receive more attention. Others would proceed in silence for weeks, never asking a question or releasing them all in a deluge, all at the same time. Giving every student an individual, detailed, timely appraisal would demand an unbearable effort from the teaching team… and, come to think of it, from the students too: if we are providing public feedback, imagine having to read individual feedback addressed to that many classmates!

AI has altered this state of affairs. It has made it feasible to ask for more concrete realisations, create short feedback cycles for everyone, and devote teaching attention to those aspects where lived experience, knowledge of the class, a reading of the person and the situation, and professional judgement really add something.

I hope it is clear that this context does not belong to computer science alone. Many fields face the challenge of getting complex concepts, techniques, and principles to be genuinely understood, mastered, and applied (”mobilised”, as my Brazilian colleagues so endearingly put it). Project-based learning is common in such situations; this case in object-oriented programming just happens to be my own circumstance.

Two weeks with an idea that fights back

The semester project is not assigned by the instructor. The student proposes it, because the student has to want to make it happen. If the instructor chooses it, all the criteria concerning what matters vanish; they become an illusion, because the real criterion becomes “*please the teacher*” or “*guess what the teacher wants*”. Everything therefore begins with a proposal from the student, while the teacher provides only qualitative contours. I asked for personal or professional relevance, for a project rich in structural decisions, and for something that could fit the available time. “*I’ll be making an application*” is not enough. “*I’ll make an innovative application to improve the user experience*” says even less by faking it too much. The student needs to discover what they really want to lay bare, what they want to make happen, and why it matters to them.

During the first week, the proposal goes through several forms.

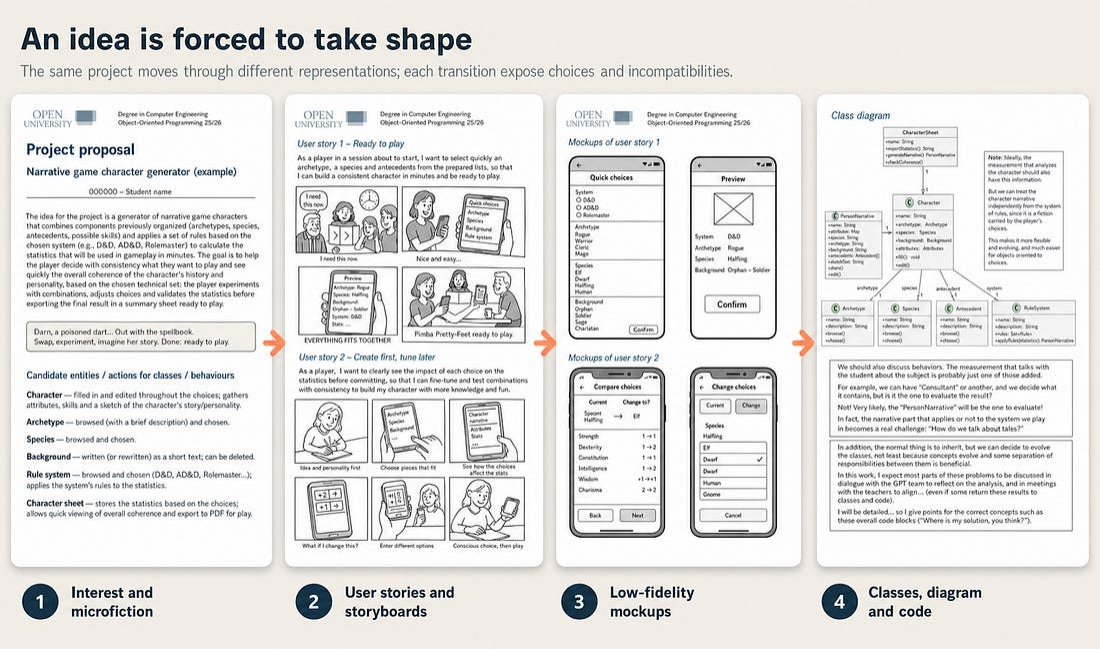
It begins with just one paragraph setting out the project idea. A short one! No Faulkner-length paragraphs (usually hollow in intention; it was already like that before, and is even more so in the age of GPTs). That idea must immediately be made concrete in a 25-to-35-word

micronarrative, produced through a structured approach called "micro SciFi-Prototyping" (μSFP, a technique I learnt from Vic Callaghan), which requires thinking about characters, situations, and actions. It is the first of many conversions of the idea. The second changes to an analytical view: what entities exist within the idea, and what behaviours do those entities have, or what behaviours exist between them? The cognitive conversions continue. The student then reveals one or two clearer narratives showing how they imagine their project taking shape in real situations: still very short things, which computer scientists call *user stories* (a rather naff piece of jargon; people probably think it sounds "more technical", but it is the term in use). Then another conversion of the idea! The student represents them as visual comic-strips (the posh name is *storyboards*) and draws one or two sketches (the term is "*mockups*" or "low-fidelity prototypes") of what the screens in the computer applications used in those narratives might eventually look like.

The drawing may be crude: aesthetics are not being assessed. It may also be beautiful; that is for each person to decide. What matters is whether the situation in this visual narrative has really been imagined, whether it is credible and pertinent, whether it makes sense in relation to the original idea as clarified in the micronarrative, and whether the screen sketches reflect what they need to allow to happen in the narrative's storyboard. With each successive cognitive transformation, the idea finds fewer places to hide inside each person's mind: well-intentioned sentences start leaving the ideas' toes and bellies sticking out, while futile adjectives collide with the real world revealed through narratives and sketches.

During the second week, students begin to establish the cognitive connection with the course content, which they had been asked to study from the outset. They must identify the first candidate "classes", draw a simple diagram of them, and produce small prototypes in Python code. The assignment brief deliberately limits the scale of what

is submitted: two or three small classes, one or two methods beyond the constructor (reader: if you do not know what that is, never mind), two or three instances (reader: if you do not know what those are, make a note saying “another technical concept”), and about sixty lines of code. The student does not have to submit everything they produce, because in an environment where AI is available, volume demonstrates no value. They do have to choose examples that are relevant for the purposes of feedback and assessment. We need only enough material to confront all the elements with one another, see whether they are coherent, and find whether they reflect a personal interest, a personal choice, and so forth.



The same project is expressed in several ways. These images come from the example provided to students; they are not student work.

This is quite a literal implementation of something Seymour Papert and Idit Harel associated with constructionism: building knowledge while building an entity that can be subjected to the “test of reality”. Does it hold up in reality? Can it be shown, discussed, and reformulated? In a chapter from the same book, Uri Wilensky proposed that abstract and

concrete concepts are not different kinds of things: abstraction is not a fixed property of a concept or object. It depends on the richness of the relationship we construct with it, on the representations, interactions, and connections we manage to establish. A project idea becomes more concrete when the student gives it life and brings it into the real world: when they narrate, draw, model, and program it, and find differences between those forms.

The almost inevitable conflict throughout these transformations has a disclosure function: it demands critical thought and reflection. To use the example, if a character in the idea's micronarrative can have several places where they store equipment but the class diagram reveals a single global inventory, there is a conflict. If the storyboard says the user is comparing choices but the mockup has space for only one, there is a conflict. If the student claims the inventory is encapsulated (a course concept) but the script reveals that the game master can change it directly, there is a conflict. A project idea, both in general and in its connection to the course content (object orientation), ceases to be a list of words. The student now has a problem of their own to untangle, the greatest problem of all: *what do I want?*

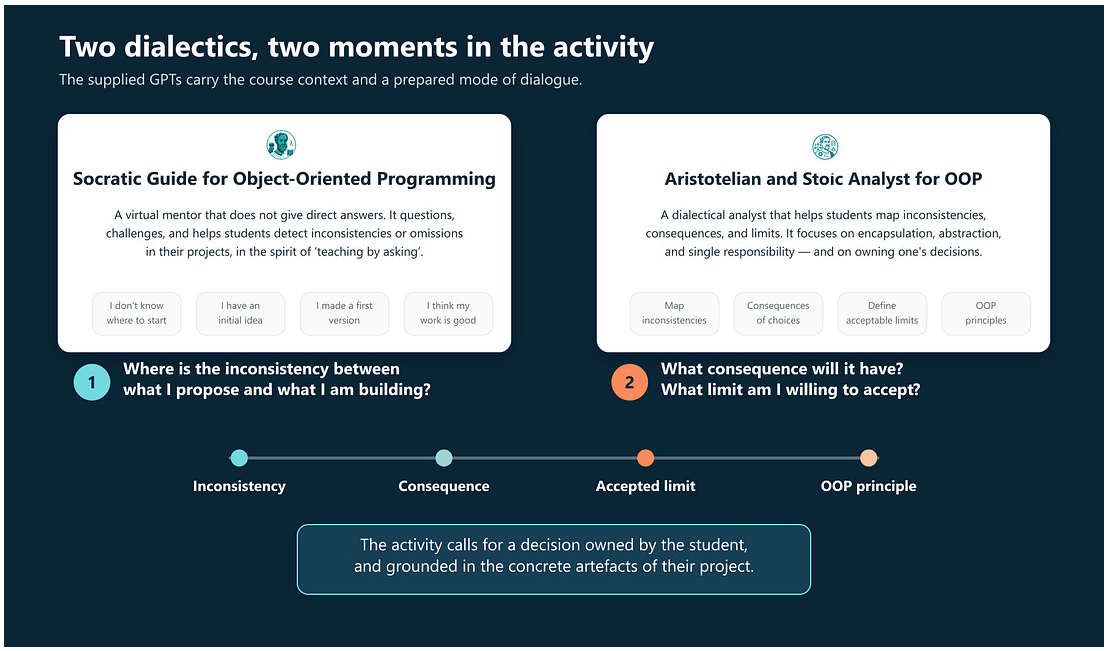
Two GPTs for structuring dialectics within an open ecosystem

Students could use any GPT throughout the course. I also provided two custom GPTs, accessible through direct links from the course pages. They were neither the "authorised" tools nor "indispensable". Their purpose was to give every student access to two modes of dialectic already structured around the learning objectives of the course, the tasks, and the object-oriented programming principles under study.

The Digital Socrates asks questions, insists, and looks for inconsistencies between what the student says they want, what they proposed, and the ways in which they have begun to give it form. It

should resist the temptation to offer a ready-made solution, although we know preventing it from doing so is impossible; it simply does not tend to jump straight there. The second GPT combines an Aristotelian and Stoic orientation. It helps explore the consequences of each choice and leads the student to decide what limits they accept for their ambitions.

There could have been three GPTs: one for the inconsistency, another for the consequence, and another for the limits. I chose two because the activity had two moments. First, discover the distance between the intention and the form it had been given in the design. Then understand what that distance will do to the project and decide how far it is reasonable to resolve it during the semester.



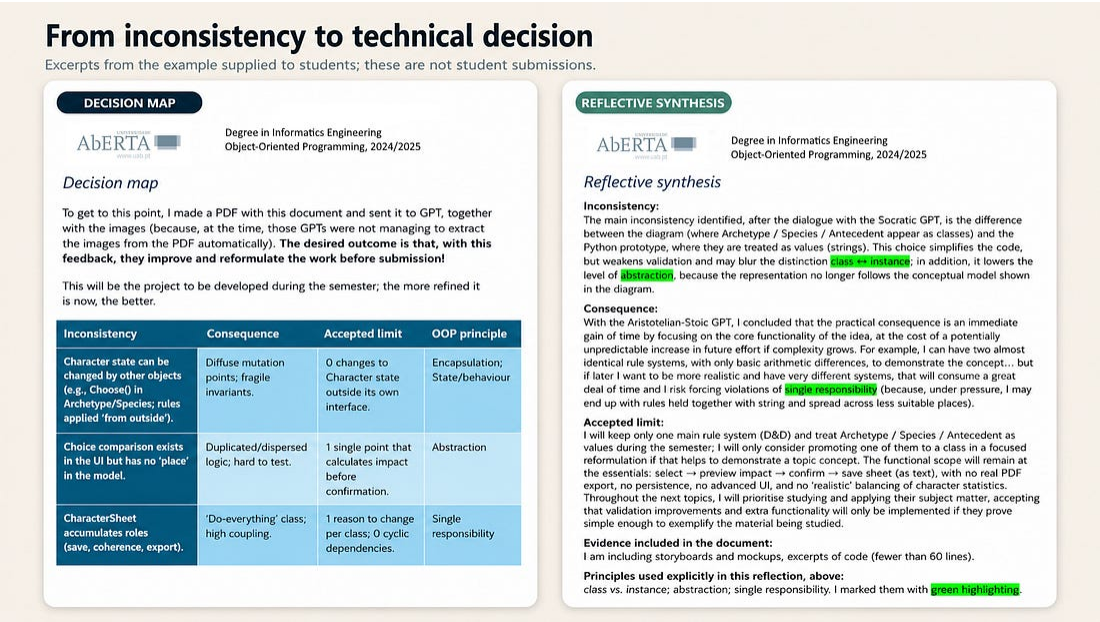
The GPTs provided bring context and a form of dialogue. The prompts used to configure them are also shared.

This last point matters. Yes, I know: if it did not, I would not have brought it up. But I am stressing it to capture your attention. An academic project does not have to be perfect. Even the meaning of

“good” varies across projects, students, and points in the semester. The available time varies, as do prior experience, the risk we are prepared to take, and the effort that is possible or desired; the course concepts surround all of this. A student may retain an option that is simpler now but will make their work harder in future. They need to see that consequence, define the bounds of the compromise, and take responsibility for the decision.

That is why the map requested during the second week had four columns:

Inconsistency → Consequence → Accepted limit → Object-oriented programming principle



From inconsistency to technical decision

Excerpts from the example supplied to students; these are not student submissions.

DECISION MAP

AbERTA

Degree in Informatics Engineering
Object-Oriented Programming, 2024/2025

Decision map

To get to this point, I made a PDF with this document and sent it to GPT, together with the images (because, at the time, those GPTs were not managing to extract the images from the PDF automatically). **The desired outcome is that, with this feedback, they improve and reformulate the work before submission!**

This will be the project to be developed during the semester; the more refined it is now, the better.

Inconsistency	Consequence	Accepted limit	OOP principle
Character state can be changed by other objects (e.g., Choose() in Archetype/Species; rules applied ‘from outside’).	Diffuse mutation points; fragile invariants.	0 changes to Character state outside its own interface.	Encapsulation; State/behaviour
Choice comparison exists in the UI but has no ‘place’ in the model.	Duplicated/dispersed logic; hard to test.	1 single point that calculates impact before confirmation.	Abstraction
CharacterSheet accumulates roles (save, coherence, export).	‘Do-everything’ class; high coupling.	1 reason to change per class; 0 cyclic dependencies.	Single responsibility

REFLECTIVE SYNTHESIS

AbERTA

Degree in Informatics Engineering
Object-Oriented Programming, 2024/2025

Reflective synthesis

Inconsistency:
The main inconsistency identified, after the dialogue with the Socratic GPT, is the difference between the diagram (where Archetype / Species / Antecedent appear as classes) and the Python prototype, where they are treated as values (strings). This choice simplifies the code, but weakens validation and may blur the distinction class ↔ instance; in addition, it lowers the level of abstraction, because the representation no longer follows the conceptual model shown in the diagram.

Consequence:
With the Aristotelian-Stoic GPT, I concluded that the practical consequence is an immediate gain of time by focusing on the core functionality of the idea, at the cost of a potentially unpredictable increase in future effort if complexity grows. For example, I can have two almost identical rule systems, with only basic arithmetic differences, to demonstrate the concept… but if later I want to be more realistic and have very different systems, that will consume a great deal of time and I risk forcing violations of single responsibility (because, under pressure, I may end up with rules held together with string and spread across less suitable places).

Accepted limit:
I will keep only one main rule system (D&D) and treat Archetype / Species / Antecedent as values during the semester; I will only consider promoting one of them to a class in a focused reformulation if that helps to demonstrate a topic concept. The functional scope will remain at the essentials: select → preview impact → confirm → save sheet (as text), with no real PDF export, no persistence, no advanced UI, and no ‘realistic’ balancing of character statistics. Throughout the next topics, I will prioritise studying and applying their subject matter, accepting that validation improvements and extra functionality will only be implemented if they prove simple enough to exemplify the material being studied.

Evidence included in the document:
I am including storyboards and mockups, excerpts of code (fewer than 60 lines).

Principles used explicitly in this reflection, above:
class vs. instance; abstraction; single responsibility. I marked them with green highlighting.

The map turns a vague observation into a technical decision. The synthesis connects that decision with the diagram, the code, and the other artefacts.

A generic GPT could also discuss these matters with students; I need not have provided the Digital Socrates and the Aristotelian-Stoic GPT. If the student failed to supply the context of the course and the task, however, they would not obtain an appropriate interaction. The

disparity in the quality of those interactions would be enormous across a large group of students. The prepared GPTs lower that barrier. There is also an institutional and data-protection issue. In light of the General Data Protection Regulation (GDPR, well-intentioned but ill-fated), it is unacceptable to require every student to create an account and send personal data to a service outside the European Union merely to use a particular GPT. And if we can avoid imposing costs on students, it is right to give them that freedom. Use of the GPTs I created as OpenAI *custom GPTs* is therefore optional. In a Prompt Grimoire within the course, I publish the prompts for every GPT I use. I also publish their reference files whenever necessary, allowing students to reproduce the interaction in another system. As alternatives to OpenAI, they may use a service hosted in the European Union, in Portugal, at the institution itself, from another provider of the student's choice, or even a language model running on their own computer. Tools such as Ollama and LM Studio already make it possible to run models locally without sending conversations to an external service.

Automated feedback for everyone changes the teaching work

The logistical gain from accepting and encouraging the use of GPTs is immediate: every student can obtain detailed, personalised, specific feedback without waiting for an instructor to find the time to get to their project and analyse it in full depth. Yet the presence of GPTs in the course creates a permanent demand upon instructors and tutors… and students. We all have to demonstrate our value beyond what the GPT already provides.

This has interesting consequences for forum dynamics. When faced with a question, I can reply: "This is what the Digital Socrates says when I put your question to it." And then add: "And this is what I have to say on my own part." The first response will usually address formal aspects or generic or canonical expectations: an omission, a criterion, an example.

The second may recognise a particular intention, relate it to something that happened in another project, distinguish a conceptual difficulty from a reasonable decision, notice that the student is steering clear of an important risk, or say, from experience, where it is worth persisting.

This open use of AI encourages students to share their AI conversations, questions, and results. After all, if "*even the teacher does it…*", the conversation no longer remains hidden for fear that it might seem illegitimate. No, colleagues, telling students that they may use it is not enough, because years of hard experience have taught them that teachers do not always practise what they preach. It matters to do it as well as say it. Their classmates can intervene in the same way. They may compare forms of prompting, challenge an answer, or realise that they face similar problems. And a student need not remain silent while waiting for the instructor to validate every small step.

Teaching work does not disappear! It becomes more exposed. If my contribution amounts to the same as the GPT's, or is even worse, its uselessness becomes evident. The richness of lived human experience has to appear in what we choose, what we value, our reading of the situation, and the way we help this person and this class move forward.

What went well

The first result was establishing feasibility. A process that previously could not even have been requested, because it would have taken several weeks and distributed support very unevenly, fitted within the effort allocated to two weeks. Students had access to immediate and more detailed feedback than the teaching team could individually have provided to more than two hundred people.

The topics proposed were rich and varied. Requiring personal or professional relevance gave the projects a direction that a standard list of topics would not have provided… and where it did not, this allowed us to alert students to the problem early (at the end of those two

weeks). The multiple concrete realisations allowed that direction to be discussed, rather than remain hidden, implied by an initial statement no one would ever examine again.

The diversity of perspectives in the class became visible, as did the cloning of ideas. Different projects make the same principles stumble in different places; identical projects following identical progressions expose a lack of authenticity. Abstract course concepts such as “single responsibility” seem self-evident in an example prepared for a class, but in a personal project what counts as a “single responsibility” becomes debatable and needs to be defended.

These examples are merely operational indicators, observations I made. They do not yet show that students learnt more or learnt better. They show that it was possible to provide the general body of students with a kind of journey and feedback that, at this scale, would previously have been reserved for a few—or might not even have been attempted.

What went wrong and was useful for that very reason

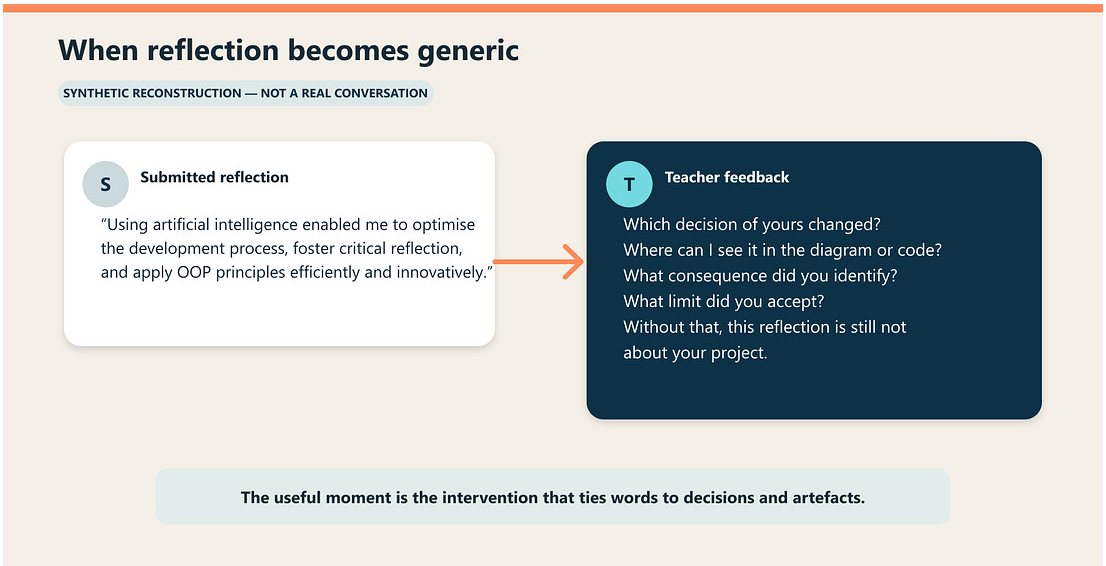
Above, when describing what went well, I already included problems: students who proposed projects without personal relevance, and cloned ideas, as I described them, because they had been requested from a GPT without introspection or copied from classmates. That counts as “going well”: a process that reveals problems earlier rather than later is a good process, because it allows timely corrections. In this section, addressing what went wrong, I look at the process itself rather than individuals. These are the aspects that prevent the intended value from emerging.

What troubled me most was having naively included a request for a final synthesis. In that request, I deliberately left room for a less structured voice. I wanted students to explain, outside the straitjacket of a table, the inconsistencies they had found, their consequences, their accepted

limits, and the relationship with the principles covered by the course. Sometimes that was what they did. In many submissions, something else happened: generic text, full of the right vocabulary and probably produced by a GPT without personal appropriation. “Optimisation”, “critical reflection”, “innovative approach”, “efficient application of principles”. The words were there; the personal project and the individual’s perspective had disappeared.

OK, you will say: then isn’t it also good that this was revealed early? No. In a generic project, the student may believe that a personal touch is unimportant and that any topic will do for learning. There is some room to believe that they are deluding thenselves into thinking they are doing things well (when they are in fact not); here, however, no student can be in any doubt that they are evading reflection.

What went wrong was that I had not planned my assessment approach with this in mind. I should have. I only did so halfway through the process, once I became fully aware of what was happening. If I assess this text as I would assess a reflection assumed to have been written without AI, I enter a spiral of frustration. I have to ponder whether an intention lurks behind a generality or whether there is only emptiness. I have to wonder whether that lingo (e.g., “trade-off”, “tension”, “signal”) came mindlessly from the GPT, or whether the student liked the idea and did not mind the term. Broadly speaking, I find myself producing feedback for an effort that never took place. In these cases, which are particularly vulnerable to empty rhetoric, the focus of assessment must go straight for the jugular: does this carry personal meaning? Is there a purpose that clearly belongs to the person proposing the project? Is there a decision? Are specific aspects of the project connected with the experience of making it? Is there agency? The feedback only moves on if that is evident; otherwise that is what we discuss, and nothing else.



Synthetic reconstruction: it does not reproduce any real student dialogue or work.

A vague formulation, or one made of buzzwords, should receive unequivocal criticism. That initial feedback sets the tone for the rest of the semester. It shows that the criteria were not decoration on a rubric and clarifies what I mean by reflection: establishing relationships that only this person can establish between their intention, their choices, and what they have built.

The same thing occurred, to a lesser extent but still far too often, in the tables of inconsistencies, consequences, limitations, and adoption of course principles. This likewise required a reorganisation of the approach to assessment and feedback, so that it would focus on these qualities as a prior condition, before exhausting itself in discussion of empty words.

There is also work to be done here on the instructional design. Perhaps these attitudes can be reduced through counterexamples alongside examples of how to carry the activity out. Perhaps GPT feedback can be improved so that it warns students about these defects in advance,

rather than waiting for the irreversible moment of human assessment. Perhaps there are other viable approaches.

Cute ideas only become serious when we write the assignment brief

The Object-Oriented Programming case offers no recipe for every field. But I hope it may be useful in many! It shows an approach that can be adapted: begin with something that matters to the student; force the idea to pass through different concrete realisations and varied cognitive transformations; use AI to provide frequent interlocution; look for inconsistencies between intention and realisation; analyse the consequences of those inconsistencies; decide which limits are accepted in resolving them; focus the instructor's intervention on critical thinking, purpose, individualisation, and the relationship between this person and what they are building.

This is how an apparently simple task—proposing a project topic—came to occupy two weeks without turning into curricular sausage-stuffing. Proposing a project topic became part of the learning itself. The student began to encounter object-oriented programming inside the problem they had chosen to build, before the project was large enough to hide its decisions beneath hundreds of lines of code.

In the previous essay, I ended by asking what process leads a student to value, decide, revise, and assume responsibility. This was one possible answer, applied under real conditions and still imperfect.

This is also where pedagogical guff has to become the trumpets of reality. Once we have championed understanding, appropriation, and critical thinking, we have to decide what happens during the first week, what is submitted in the second, what feedback reaches everyone, where the instructor comes in, and what we do when impeccable prose contains no soul inside it.

References and links

- Papert, S., & Harel, I. (1991). *Situating Constructionism*. First chapter of *Constructionism*.
- Wilensky, U. (1991). *Abstract Meditations on the Concrete and Concrete Implications for Mathematics Education*. In I. Harel & S. Papert (Eds.), *Constructionism*, pp. 193–204.
- European Data Protection Board. International transfers and international cooperation.
- Ollama and LM Studio, options for running models locally.