Fifteen Principles for the Dev-sumer Computational Designer

1. Tools Are Not Neutral

Every piece of software embeds assumptions about how work should be done. When you accept a tool uncritically, you accept those assumptions—including toxic ones about overtime and productivity. When you build your own, you encode your own values: efficiency, humanity, respect for time.

2. The Boundary Between Designer and Tool-Maker Is Artificial

This separation made sense in the industrial age when tools required factories. It makes no sense in the computational age when tools require thinking. The same logical thinking that makes you a good designer makes you capable of building tools.

3. Building Tools Is Design Research

When you build a tool, you're forced to formalize your thinking about the problem. This clarity makes you a better designer, even when not using the tool. You need to be a problem maker before being a problem solver—understand deeply, then solve elegantly.

4. Start With the Problem, Not the Tool

Don't ask "What can this software do?" Ask "What does this problem need?" Then build (or find, or modify) tools that address that need. The tool serves the vision, not the other way around.

5. Automate Toil, Not Thinking

Build tools that eliminate tedious, repetitive, soul-crushing work so you can focus on creative, strategic, meaningful thinking. Don't build tools that do your thinking for you—build tools that give you time to think.

6. Make Your Tools Shareable

Knowledge grows when shared. Your custom components, scripts and workflows are gifts to the community. Share them. Document them. Let others build on your work. Use commons licenses (like Ladybug Tools does) to encourage this.

7. Embrace Imperfect Tools Over Perfect Workflows

A tool that works 80% and saves you hours is better than a manual process that's 100% but takes days and crushes spirits. Perfect is the enemy of shipped. Iterate in practice, not in theory.

8. Learn Enough to Be Dangerous

You don't need a computer science PhD to build tools. Learn enough Python, C#, or JavaScript to solve your immediate problems. Depth comes with practice. I learned algorithms in engineering with MATLAB and C++, but the logical structure was what mattered—computational thinking, not syntax.

9. Build Iteratively, Not Monumentally

Don't try to build the perfect tool from day one. Build something that solves today's problem. Improve it tomorrow. Let it evolve with your understanding. (AN.: The Ambrosinus Toolkit grew component by component, problem by problem).

10. The Tool Is the Message

When you share a tool, you're not just sharing code. You're sharing a way of thinking about problems. You're teaching by demonstrating. The tool is pedagogy—it shows how you structure thought.

11. Leadership Must Evolve With Technology

An effective manager must know how to recognize and value the uniqueness of each team member. There is no universal approach. Each professional has their own way of expressing talent, their specific competencies, their own key to reading the project. The true ability of a leader lies in knowing how to tune these different instruments into a harmonic design symphony.

12. Protect Human Time Fiercely

Time is the only resource you can never get back. Tools that save time aren't just about efficiency—they're about dignity. They're about evenings with family, weekends for rest, mornings for reflection. Don't normalize the sacrifice of life for work.

13. Combine Disciplines for Creativity

Combinatorial creativity comes from interacting with many people and many operating environments. Engineering + Architecture. Programming + Design. Environmental science + Parametric tools. AI + Human judgment. The intersections are where innovation lives.

14. Stay Human in the Age of AI

With the rise of artificial intelligence, some observations may become obscured or forgotten while others may be validated. In this period of cultural uncertainty, it is essential to remember the human element. Architects and engineers are believed to be among the last intellectuals within the technocratization of the AEC industry, capable of offering reflection, observation, and direction beyond productivity.

15. Cultivate Passion and Curiosity

Passion is the inner flame that transforms work into vocation, that makes you rise at dawn to pursue solutions others haven't imagined. Curiosity is the spark that whispers "what if?" and "why not?"—the engine that drives you to explore unknown territories and find unexpected connections between distant ideas. When passion and curiosity merge, they create extraordinary synergy. Passion provides the determination to pursue the questions curiosity raises, while curiosity continuously fuels passion with new challenges and perspectives. Together, they allow you to grow continuously, reinvent yourself, and leave a significant mark—in your studio, your firm, your discipline, or the world. These qualities have no age limit. They are gifts you can cultivate throughout your life, transforming each day into an opportunity for discovery and every challenge into an occasion for growth.

LUCIANO AMBROSINI ARCHITECT

Adapt | Share | Build | Acknowledge