Outlast the Tools

A Game Producer's Field Guide to AI

Production craft doesn't change when the tools do.

Rob Sandberg

Outlast the Tools

A Game Producer's Field Guide to AI

Rob Sandberg

Sample Edition

Introduction: Thirty Years and One Honest Admission

It started as an annotation problem.

I had been working on a game project by myself for a few months, the kind of parallel-track development that producers accumulate over a career: personal, unhurried, but real. The Figma file existed. A design document existed. The gap between them was the problem. A set of UI screens needed specification annotations, and the notes that should have fed them were scattered across my own files: a running document of design thinking, half-formed decisions, and several months of incremental choices I had not yet assembled into anything coherent. I was planning to show the project to someone and wanted the specification in proper shape before I did.

I opened Claude Code. [EN: Claude Code runs in a Mac or Windows terminal rather than a browser. Setup requires installing a few packages via the command line, which looks more intimidating than it is. I am not a developer. I got it working by asking Claude Code to walk me through each step.]

Not the chatbot. Claude Code is a command-line tool that runs AI agents directly in your working environment: against your files, your local tools, your actual project. The distinction matters. A chatbot answers questions. An agentic tool does work.

The first prompt was narrow: help me annotate this UI screen from these notes. The model asked a clarifying question. I answered it. The annotation came back structured, specific, and about 70 per cent right. I corrected the 30 per cent and pushed back into the session with more context. The second iteration was better. The third was almost exactly what the screen needed.

And then I stopped accepting the narrow version of what I was doing.

The annotation session became a living specification system. The Figma screens became the source of record, with annotations generated from the accumulated context in the session rather than from a separate attempt to reconstruct each design decision. I added a second agent working on the Confluence integration, so the annotations pushed into a documentation hub as they were written. I added a third on the Jira side, mapping annotation items to acceptance criteria. By Saturday evening, what had started as twenty Figma annotations had become something closer to a complete specification layer for the UI: consistent, linked to the work tracking system, derived from the actual design rather than from notes about it.

By Sunday I had added a Figma export pipeline, a SQLite schema to hold the structured data across sessions, and what I can only describe as an orchestrated team of more than twenty specialist agents, each with a specific scope and a constraint on what decisions they could make without my explicit sign-off. The database schema meant that the context was persistent, not session-dependent. Each agent could see the relevant history of decisions without me reproducing it in every prompt.

When I showed the project, the person I showed it to asked who else had been working on it. The documentation looked like the output of a full production team. I had built it alone, over a weekend.

The Honest Admission

What made the weekend work was not the tools. The tools are genuinely good. What made it work was the knowing: what the annotations needed to say, what the Jira acceptance criteria needed to specify, what information would be missing from a Confluence page that was assembled without a production eye on it. Every time an agent produced something that was almost right, my evaluation of what was wrong was instant and specific. Every time I needed to decide what to ask for next, the decision was informed by two decades of watching specifications fail in exactly the ways that produce expensive mid-sprint surprises.

The honest admission is this: I could not have done what I did that weekend in 1995. Not because the tools did not exist (they did not), but because in 1995 I did not know enough about what a good specification actually requires to evaluate what I was getting.

The tools did not make me a better producer. Thirty years of being a producer made the tools useful.

This is the argument the rest of the book makes, and everything in it follows from this premise. AI handles the volume. The producer handles the judgement. That distinction is not a consolation offered to people worried about their jobs. It is a description of what actually happened, in practice, across everything documented here. The AI

produced the first draft, every time. The first draft was never the answer. What made the answer was the judgement about which 30 per cent was wrong and what to do about it.

Who This Book Is For

This book is written for the producer who is already using AI tools, or who is about to start, and who is finding that the available guidance splits into two unhelpful camps: breathless enthusiasm that doesn't match her daily experience, and defensive dismissal that doesn't help her figure out what to actually do.

The reader I have in mind has shipped at least one game. She understands scope management, stakeholder relationships, sprint retrospectives, and the specific social dynamics of milestone reviews. She does not need to be taught what a backlog is. She needs to understand how AI changes the work she already knows how to do, and how to use it without compromising the judgement that makes her valuable.

This book also has something to say about producers who are earlier in their careers and are entering a game industry that is adopting AI tools faster than it is thinking about what those tools do to the formation of production judgement. Chapter 1 names this the deprofessionalisation problem. The closing bookend returns to it. The concern is not rhetorical.

What this book is not: a list of AI tools to use. The tools available in 2026 will have been replaced or transformed by the time most people read this, and a tool list that becomes outdated in twelve months is

not a field guide. The principles that govern when to use AI, how to evaluate its output, and what to keep doing yourself do not have a version number.

The Translation Gap

The producer reading this is probably not behind. That is what the available discourse implies: the tools are moving fast, the junior roles are disappearing, the studios that do not adopt will be left behind. Most producers I speak to have absorbed this framing and assumed their relative position in it is bad.

The argument this book makes is different. The skills that make a producer effective are what AI-assisted production requires to function. Maintaining context across a complex project and evaluating output against a quality standard derived from experience rather than specification: these are the governance layer the technology needs. The gap between what producers already know how to do and what AI-integrated production requires is a translation gap. Filling it is what this book is for.

There is a corollary. Producers who are early in their careers, who have not yet accumulated the recoverable mistakes that install production judgement, are in a more exposed position than the current enthusiasm for these tools suggests. The closing bookend returns to that. It is a concern this book takes seriously without claiming to resolve it.

How to Read It

The book has two parts. Part One covers the foundations: what is actually happening in the industry, what skills transfer from game production to AI management, how to build systems that outlast the specific tools you are using today, and how to protect against the legal and institutional risks that arrive with any AI adoption.

Part Two applies the framework to every major phase of game production, from concept through live operations. Each chapter takes a specific domain, names the AI capability relevant to it, and describes what the producer does that the AI cannot. The chapters can be read in sequence or used as reference for the specific phase you are currently working in.

Five transferable producer skills recur throughout: prompt precision, output taste, context architecture, iteration without attachment, and knowing when to take the wheel. Chapter 2 defines them. Every subsequent chapter applies at least one. They are the framework the book operates on.

A note on endnotes: every citation marked [EN: ...] in the text is documented in the endnote section at the close of each chapter. Several endnotes across the manuscript flag citations for pre-publication verification, where the source is real but the specific publication details require confirmation. These are marked clearly. They are not placeholders for invented citations; they are genuine sources from the GPA post archive whose formal bibliographic details are pending.

A note on terminology: this book uses Scrum vocabulary throughout (sprints, burn-downs, retrospectives), not as an endorsement of Scrum. I am on record as not being a Scrum advocate. I use the terminology because it is so widely understood in game production studios that inventing alternatives would create more friction than clarity. The underlying principles in this book apply regardless of which methodology your studio runs.

The weekend with the Figma file was not typical. Most days, working with AI is more incremental: a better first draft, a faster data pull, a search across a body of text that would have taken three hours to read. The principles hold regardless of scale. Evaluate the output. Correct the 30 per cent. Bring the judgement the AI cannot bring.

The Thing AI Cannot Do

Opening bookend, Draft 2, 2026-06-19

Every AI decision in this book comes down to two questions: does this make the work better for the team doing it, and does it make the product better for the players receiving it?

These are not productivity questions. Productivity is a measurement, and measurement answers a different question: did output increase? The questions above ask whether the humans involved are doing better work, in conditions that sustain them, and producing games that are worth playing. Those outcomes and higher output can coincide. They can also diverge. The test is not whether AI speeds things up. The test is whether the result of that speed is worth having.

Part Two applies the framework built in Part One to every major phase of game production. Before it does, this bookend states the values it operates on.

The emotional infrastructure of game production is not widely discussed. The industry teaches producers how to manage projects. It does not teach them how to manage themselves, or how to maintain the specific kind of emotional intelligence that production actually requires.

Fear, anger, disgust, sadness, surprise: a career in internal production generates all five, in specific patterns, in ways that are recognisable and rarely acknowledged. Fear is the baseline state for most experienced producers, even if they would not call it that. The milestone is public, the schedule is not moving, and the arithmetic runs in the back of the mind during the bus ride home. Anger accumulates in rooms where the right decision is obvious, the wrong decision is made by someone not accountable for the cost, and the producer manages the fallout while carrying what she could not say. Disgust is the slower accumulation of witnessing values violated without consequence. Sadness is the grief of projects that almost got there: the design that was genuinely good and died in committee, the team that was building something real until the conditions for building it were removed. Surprise, at its best, is the moment when the producer discovers she can solve a problem she has never seen before, and the realisation that a career of recoverable mistakes has become something that looks, from the outside, like judgement.

No AI system carries any of this. It cannot. The emotional weight is not a side effect of the job. It is part of the formation. A producer who has navigated fear, managed her anger constructively, maintained professional standards in the face of disgust, sat with the sadness of what gets lost, and accumulated enough surprise-become-competence to operate with genuine authority: that producer is not interchangeable with a system that processes requests at low latency and optimises for stated preferences.

Daniel Goleman's research on emotional intelligence identifies three clusters that predict professional effectiveness in roles requiring sustained collaborative judgement: self-awareness and self-regulation

(knowing what you feel and choosing how to act on it), social awareness (reading what others are experiencing with or without being told), and relationship management (the capacity to maintain productive working relationships under load). [EN: Goleman, Emotional Intelligence, Bantam Books, 1995] These are the specific capacities the emotional weight of production builds, when the conditions are right and the formation is allowed to happen. They are the specific capacities that determine whether AI output, however capable, is being evaluated by someone who can judge it reliably.

Mihaly Csikszentmihályi spent decades researching what makes human experience genuinely enjoyable. His Eight Channel Model maps the relationship between challenge level and skill level against nine mental states. The finding that matters for production: psychological flow, the state of complete absorption in which attention concentrates entirely on a task and the work produces intrinsic satisfaction, occurs only when both challenge and skill are high. Every other combination produces anxiety, boredom, apathy, or one of the degraded states in between. [EN: Csikszentmihályi, Flow: The Psychology of Optimal Experience, 1990; Csikszentmihályi, Massimini, Carli, Eight Channel Model, 1987]

Game production has structural tendencies that destroy the conditions for flow. Long feedback cycles push evaluation away from the moment of work. Unclear priorities force cognitive effort onto a question that should already be answered. Interruptions are expensive not because of their duration but because of the concentrated state they break. And micromanagement (the daily

operational reality of most producers, not an occasional failure mode) removes the sense of ownership that flow requires. A producer's job description is, at its most honest, a list of structural reasons that the people around them will not achieve flow without deliberate countermeasures. The intervention and coordination that define the role are precisely what disrupt the conditions the role is supposed to protect. Achieving flow as a producer, and protecting it for a team in an environment that works against it, is one of the more demanding things the job asks. These conditions are all within a producer's design authority. They are just not easy.

The AI-relevant version of this is not about whether AI can achieve flow. It cannot, and the question is category confusion. The question is whether AI adoption improves or degrades the conditions under which the humans making games achieve it. An AI that absorbs administrative load, freeing a designer to spend six hours in uninterrupted creative work, has improved the conditions for flow. An AI that generates a stream of outputs requiring constant low-level evaluation and correction, fragmenting a developer's attention across thirty micro-decisions per hour, has degraded them.

The test is not productivity. The test is the quality of the conditions for good work.

The pipeline problem, introduced in Chapter 1, belongs here too.

Research into AI-assisted software development found that developers working with AI scored 17 per cent lower on code comprehension tests than developers working without it. The

researchers named the finding the paradox of supervision: the tool that amplifies capability today degrades the capability needed to evaluate it well tomorrow. [EN: Primary source TBD, see Chapter 1 EN6]

For code, this is a diagnosable problem. Production judgement, the capacity to read a complex situation accurately, identify the actual risk rather than the presented risk, and act under incomplete information, does not have a test. It accumulates through recoverable mistakes: the errors that are expensive enough to teach something and cheap enough to survive. The junior producer who calls the wrong priority in triage, prepares the imperfect milestone build, sits with a senior's pushback on her definition of done. These are the formation mechanisms. They require a context where the mistake can happen and be recovered from, and a senior presence capable of making what happened visible.

The development cuts of 2024 and 2025 removed both conditions. The GDC's 2026 survey found that 28 per cent of game professionals had been laid off in the past two years, with cuts concentrated at the roles where formation accumulates: coordinators, associate producers, junior designers. That work was absorbed by seniors already at capacity. The recoverable-mistake pathway closed from above: no role in which to make the mistake, no senior presence with time to make it visible.

AI adoption is closing the pathway from below. The GDC found that generative AI is used most by executives and upper management: the professionals who already have the judgement to evaluate what AI returns. The tools are being applied most directly to the

administrative, documentation, and coordination work that used to sit at junior and mid-level. The junior producer who would have called the wrong priority in triage no longer has that work to get wrong. The tool absorbed it. The formation mechanism still exists in principle. The work it needs to run through has gone elsewhere.

The paradox of supervision compounds both. The pipeline is draining from above and below simultaneously. The question of who supervises the AI in 2036 is a formation question, and the answer depends on decisions being made now about whether the conditions for production formation are maintained or allowed to drain.

The two questions at the start of this bookend are not a constraint on what follows. They are the standard of evaluation.

Part Two demonstrates the framework across every major phase of game production. Each chapter names an AI capability and a production application. Each one can be measured against the two questions: is the work better for the team doing it, and is the product better for the players receiving it?

The field guide is designed to be used in work, not read once. The framework is also designed to be honest about what it cannot answer. The judgement about whether a specific AI decision meets the two-question test belongs to the producer in the room, with the team, in the specific conditions of the specific production.

That judgement is the thing AI cannot do.

Endnotes

1. Goleman, Daniel, *Emotional Intelligence*, Bantam Books, 1995. Three-cluster EI model: self-awareness/self-regulation, social awareness, relationship management.

2. Csikszentmihályi, Mihaly, *Flow: The Psychology of Optimal Experience*, Harper & Row, 1990. Conditions for flow: clear goals, immediate feedback, matched challenge-to-skill ratio, freedom from interruption.

3. Csikszentmihályi, M., Massimini, F., and Carli, M., "The monitoring of optimal experience: A tool for psychiatric rehabilitation," *Journal of Nervous and Mental Disease*, 1987. Eight Channel Model: flow occupies the high-challenge/high-skill quadrant; every other combination produces a degraded state.

4. Shen, Judy Hanwen, and Tamkin, Alex, "How AI assistance impacts the formation of coding skills," Anthropic, January 29, 2026. 17% lower comprehension finding in RCT with 52 junior engineers. The term "paradox of supervision" is the author's characterisation of the dynamic the paper identifies. See Chapter 1, Endnote 6.

Chapter 1: Your Expertise Is Invisible. AI Just Made It Visible.

I want to start by saying something the field has been reluctant to say: the frustration is accurate.

The apocalyptic version is easy to spot and easy to dismiss: AI as existential threat, studios complicit, automation taking everything that makes games worth making. That framing is wrong and it knows it is wrong. The quieter version is harder to dismiss, because it is grounded in something real. Working producers are trying to figure out what is actually happening and finding that most available frameworks either oversell the technology or dismiss it entirely.

The 2026 GDC State of the Game Industry report surveyed more than 2,300 game professionals. Fifty-two per cent said generative AI is having a negative impact on the industry. Two years earlier, that figure was 18 per cent. Only 7 per cent now say the impact is positive, and that number has declined every year the survey has included the question. [EN: GDC, "2026 State of the Game Industry"] Separately, the Game Developer Collective found in 2026 that generative AI usage among developers had fallen from 36 per cent in the first half of 2025 to 29 per cent in the first half of 2026, with 47 per cent of respondents worrying that AI would negatively affect game quality. [EN: Game Developer Collective / Omdia, developer survey 2026]

These are not the opinions of people who do not understand the technology. These are working developers, many of whom have been using AI tools daily, and what they are describing is not theoretical risk. It is direct experience of something not going well.

The fears behind those numbers are specific and earned. Job displacement. AI used to justify headcount reductions that leadership was looking for reasons to make anyway. A production culture beginning to confuse automation with quality. Tools that generate confident-sounding output without reliable grounding. Every one of those concerns deserves the attention it is getting.

The concerns are accurate. What they lack is precision. Naming the problem precisely is the first step toward doing something useful about it.

The genuine AI capability is real enough to be worth taking seriously and inflated enough to be hard to calibrate. AI can hold correlations across large bodies of qualitative data that no human can read simultaneously. Five hundred bug reports, three months of retrospectives, a year of player feedback: the connections across those are findable in minutes rather than months. It can absorb administrative overhead that has historically consumed most of a producer's working week. These are real gains. But the claims layered on top of them are not grounded in what the tools currently do. AI replacing producer judgement, automating creative decisions, running a production without a human who understands what she is supervising: none of this is what AI does today. Both the genuine

capability and the inflated claim arrive in the same pitch deck, the same conference panel, the same product announcement. That is why producers cannot tell the difference.

The problem has two parts. The first is deprofessionalisation: the process by which AI, like the rounds of layoffs before it, is making the invisible work of game production simultaneously more visible and more contested. The second is context orphaning: a new category of risk in which the reasoning behind AI-assisted decisions disappears when the session ends, hollowing out institutional knowledge in a way that game studios are particularly ill-placed to absorb.

Both of these are producer problems. Both of them are, as it turns out, problems that producers are structurally better placed to address than anyone else working in a game studio right now. That argument takes the rest of this book to make. This chapter makes the case that the moment is real enough to be worth taking seriously.

What Deprofessionalisation Actually Means

Ryan K. Rigney published a Substack post using the word *deprofessionalisation*, and it landed like a fire alarm in a building where everyone had already been smelling smoke. He argued that the jobs most at risk were those "that seem replaceable to management (even if they're not)." [EN: Rigney, "The Games Industry Is Deprofessionalizing"]

That phrase does more work than it appears to. The jobs that *seem* replaceable to management are a specific category. They are not the technically complex roles whose outputs can be measured in lines of

code or assets delivered per sprint. They are the jobs whose outputs are hardest to quantify: the producer who keeps a production from going sideways by catching a risk six weeks before it materialises; the writer who spots a story logic problem in week three rather than letting it become a rebuild in week forty; the QA lead who carries the institutional memory of the last three times a particular system broke.

These are craft roles. The knowledge inside them is real. But from the perspective of a spreadsheet two levels removed from the work, they look like overhead.

I have watched this pattern play out across every cycle of cost-reduction in game production. Every time a studio goes through that exercise, the invisible roles take the first cut. And every time, the people making the decisions are surprised by what is missing six months later. The surprise is structural: the value was invisible to them by definition. You cannot defend something on a spreadsheet if nobody has ever put it on a spreadsheet.

The GDC's 2026 report found that 28 per cent of game professionals surveyed had been laid off in the past two years, rising to 33 per cent for US-based respondents. [EN: GDC, "2026 State of the Game Industry"] The cuts were concentrated in exactly the roles Rigney identified. The result was organisational blindness. Studios that cut craft roles found that the gap left behind was not visible until something specific went wrong. The story coherence problem arrived in week forty as a rebuild. The system failure arrived as a critical bug in certification. The production risk arrived as a delayed milestone. All of these were predictable. The capacity to predict them had walked out the door.

The uncomfortable part of the deprofessionalisation argument, as Rigney acknowledges, is that some of what was cut deserved to go. The big studios had become machines: rigid, layered, risk-averse rather than efficient. Approval chains that turned simple decisions into month-long exercises in calendar management. Process that existed to protect the structure rather than the games. Craft and bloat genuinely look the same from a sufficient distance, and a great deal of what got cut was genuine bloat.

The problem is that the people making the decisions often could not tell the difference. The decision-making apparatus installed to replace what was cut carries the same blind spot. Metrics dashboards, automated status updates, AI-generated reports: they can measure what was already being measured. They cannot see what was never being measured.

This is where AI enters the picture. AI is now applying the same pressure that layoffs applied, but from a different direction. Where layoffs asked "can we justify the headcount?", AI is asking "can we automate the output?" In both cases, the logic operates on what is visible: the deliverable, the document, the artefact produced. In both cases, the logic is blind to the judgement behind the visible output.

Here is the thing that matters, and the thing this book is built around: the argument that AI can replace producer judgement is, in itself, proof that producer judgement exists. You do not set out to replace something invisible. You replace something you have identified and decided you no longer want to pay for. The fact that AI adoption has forced studios to name and scope producer expertise, even if only to argue that AI can do it, is a new development. It cuts both ways.

AI is making invisible work visible by trying to do it. The translation between creative intent and executable plan, the reading of a team's actual capacity against a plan's theoretical demands, the sense of where a project is actually going versus where it says it is going on the status dashboard: all of it was always load-bearing. It just was not legible to anyone who had not personally experienced what happens when it breaks.

That legibility is now contested space. Whether producers occupy it or cede it is the central professional question of the next decade.

The Session Nobody Saved

There is a second problem running alongside deprofessionalisation, and it has received far less attention. I have named it context orphaning, and it is a risk category specific to agentic AI tools that most game studios have not yet recognised.

When a developer works with an AI coding assistant, or any agentic AI tool, she produces two artefacts. The first is the obvious one: the code written, the document produced, the feature shipped. The second is the reasoning: the decisions made during the session, the trade-offs considered, the alternatives rejected, the specific context that shaped why the output took the form it did.

Almost nobody saves the second artefact. Almost everyone loses it.

In traditional development, some of this reasoning lives in commit messages, code comments, and the conversations attached to pull requests. The rest lives in people's heads, but at least those heads stay

attached to the same studio for a while. When a developer leaves, there is attrition. But the surrounding system retains something: the codebase she worked on, the documentation she wrote, the colleagues she influenced.

Agentic AI development does not work this way. The session is the container. The reasoning exists in the session or it exists nowhere: why the AI was instructed to approach the problem as it did, what was tried and rejected, what the constraints were. End the session and the reasoning ends with it. What remains is the output: code that works, or a document that was approved, or a plan that was followed, stripped of the context that would allow anyone else to understand, replicate, or meaningfully modify what was built.

It matters particularly badly in game studios.

Game studios carry their institutional knowledge in people: the relationships between people who have shipped things together, the unwritten understanding of why the combat system works the way it does, the producer who knows that the third-party physics middleware has a known issue that bites in the third week of integration and has been bitten by it twice before. Documentation is secondary. Code comments are secondary. That knowledge lives in heads. The studio's structural response to this has always been to try to keep the heads around long enough to transfer what is inside them.

High turnover was already endemic in games before the recent rounds of cuts. The project-based structure of AAA production burns through people at a rate that every postmortem acknowledges and no studio has solved. Agentic AI tools, adopted without a preservation discipline, compound this from a different direction. The institutional

memory that used to leave when a developer left was at least recoverable in principle: buried in the codebase, reconstructible from the documentation, available from the colleagues who worked alongside her. The reasoning from an AI session is gone when the session closes.

There is a concrete illustration of how this compounds in practice. Claude Code Game Studios, an open-source agentic coding system that attracted sixteen thousand GitHub stars in its first week, was built as a production-grade AI coding framework for real game projects. Its architecture is sophisticated: a three-tier hierarchy, 49 agents, 73 skills, 12 hooks, and 41 templates. But a close reading of how it handles institutional memory reveals the structural problem. Session reasoning is stored in markdown files. The notes that agents are supposed to carry forward between sessions, the architectural decisions, the context that explains why a particular approach was taken: all of it lives in files that agents write to and read from. [EN: Claude Code Game Studios, GitHub 2026]

Markdown files are neither persistent nor active. They have all the problems that documentation has always had in game studios: most accurate when written, progressively less accurate as the code evolves around them, and abandoned entirely when deadline pressure arrives. The session reasoning that matters most is precisely the reasoning most likely to be lost, because capturing it requires discipline that feels like overhead exactly when the pressure to move fast is highest.

The developers working on the most sophisticated agentic systems in the industry are aware of this problem and have not solved it. Producers are the natural owners of the solution, for the same reason producers are the natural owners of every knowledge management problem in game production: someone has to care about what the whole team knows, and that responsibility has never sat anywhere else.

Who Supervises the Supervisor?

Deprofessionalisation and context orphaning do not operate independently. Together they produce a structural consequence the industry has not yet fully articulated.

Research into AI-assisted software development has found that developers working with AI scored 17 per cent lower on code comprehension tests than developers working without it. The researchers called this the paradox of supervision: the tool that amplifies capability today may degrade the capability needed to use it well tomorrow. [EN: TBD, AI code comprehension study, "paradox of supervision"]

For code, this is a problem with a measurement mechanism. You can test for code comprehension, run assessments, and adjust. Uncomfortable, but addressable.

Production judgement does not have a test. It is the capacity to read a complex project situation accurately, identify the actual risk rather than the presented risk, and act under incomplete information. It accumulates through experience, specifically through what I think of

as recoverable mistakes: the errors that are expensive enough to teach something but cheap enough to survive. Call the wrong priority in a triage meeting. Miss a dependency in a schedule. Prepare a milestone build that gets pushed back by a lead who tells you your definition of done is wrong. These are the mechanisms of formation, and they require two conditions: a context where the mistake can be made and recovered from, and a senior presence capable of making what happened visible.

Both of those conditions are being removed simultaneously.

The GDC cuts were concentrated at junior levels: coordinators, associate producers, junior designers. The work of those roles got absorbed by seniors already operating at capacity. The recoverable-mistake pathway closed from above. The junior producer who would have called the wrong priority no longer has the role in which to call the wrong priority.

At the same time, AI is absorbing an increasing share of the work that used to sit at junior level: scheduling support, documentation, status reporting, research, brainstorming. The GDC report notes that the game professionals most likely to be using generative AI are business professionals and upper management, the people who already have the judgement to evaluate AI output and catch its mistakes. [EN: GDC, "2026 State of the Game Industry"] The tool is being adopted most enthusiastically by the people who need it least for developmental purposes, and applied most directly to the work that used to build the developmental pipeline for everyone behind them.

The result is a pipeline broken from both ends. Junior roles are being eliminated from above by redundancy and from below by automation. The roles that survive do not generate the recoverable mistakes that produce senior judgement, because seniors absorbed the work and AI absorbed the rest.

The question this raises is not abstract. Who supervises the AI in 2036? Specifically: where are the producers with decades of recoverable mistakes behind them, the ones who can look at an AI-generated production plan and know, from the scar tissue, exactly where it is going to fail?

That cohort is being trained right now, or it is not. The conditions that produce it are either in place or they are not. What the data in this chapter suggests is that those conditions are not in place. The industry is managing immediate disruption. The developmental pipeline is quietly draining.

This is the hardest version of the deprofessionalisation argument. AI is making expert judgement contestable now. It is also, combined with the structural changes that arrived alongside it, making it difficult to produce the expertise that would allow anyone to evaluate AI output with competence later. The two problems compound each other. The compounding is structural and invisible on a spreadsheet.

Endnotes

1. GDC, "2026 State of the Game Industry," January 2026. 2,300+ respondents. 52% of developers say generative AI has a negative impact (up from 30% in the 2025 report and 18% in the 2024

report). The GDC 2026 press release states verbatim: "up from 30% last year and 18% the year prior." The manuscript's "two years earlier" figure of 18% refers to the 2024 GDC report and is confirmed accurate.

2. Game Developer Collective / Omdia, developer survey, H1 2026. North America and Europe. Usage figures: 29% H1 2026, down from 36% H1 2025.

3. Rigney, Ryan K., "The Games Industry is Deprofessionalizing," Push to Talk (Substack), April 18, 2025. pushtotalk.gg/p/the-games-industry-is-deprofessionalizing. Note: title uses lowercase "is." The post predicts marketing roles would be first, followed by "roles that seem replaceable to management (even if they're not)."

4. GDC, "2026 State of the Game Industry," January 2026 (as EN1). 28% of game professionals laid off in past two years, rising to 33% for US-based respondents.

5. Claude Code Game Studios, open-source agentic coding framework, GitHub, 2026. 16,000 stars, 2,400 forks in first week. **Add full attribution (author/organisation) before publication.**

6. Shen, Judy Hanwen, and Tamkin, Alex, "How AI assistance impacts the formation of coding skills," Anthropic, January 29, 2026. RCT with 52 junior engineers learning a Python library (Trio). AI-assisted group scored 50% on a follow-up comprehension quiz; hand-coding group scored 67%, a 17% gap. The term "paradox of supervision" is the author's characterisation

of the dynamic the paper identifies; the phrase does not appear in the paper itself. Referenced in GPA post "Who's Going to Supervise the AI?"

7. GDC, "2026 State of the Game Industry," January 2026 (as EN1). Most positive group: executives and business operations (19%).

Chapter 2: Engineers Had to Invent What You Already Do

In February 2026, Harvard Business Review published a piece describing a new management role it believed AI-era organisations urgently needed. The role involved setting clear context for AI systems, translating business intent into actionable briefs, evaluating output quality against criteria the system itself could not generate, and managing iteration loops between AI capability and organisational need. The article framed this as an emerging discipline, one that few people understood and that most organisations were scrambling to staff. [EN: HBR, "AI orchestration role article", February 2026]

Producers who read it recognised their job description. The framing was new. The work was not.

Engineers building agentic AI systems have arrived at the same place by a different route. The most sophisticated open-source agentic coding framework published in 2025 built its governance layer around a protocol it called Q→O→D→D→A: Question, Options, Decision, Draft, Approval. The producer agent within that system is explicitly forbidden from making decisions. Its defined function is to price trade-offs, surface options, and hold the decision point open until a human closes it. [EN: Claude Code Game Studios, agentic coding framework, GitHub 2025]

The Q→O→D→D→A protocol is a rediscovery: it describes what a competent producer does in sprint reviews, milestone gates, and creative reviews. The engineers who built it arrived at it by working out what they needed an agentic governance layer to do, finding that it required human decision authority at every meaningful junction, and building the protocol from first principles because nobody had told them it already existed.

The claim this chapter makes is that the same pattern applies across the whole domain. The skills game producers carry, running collaborators of varying capability, managing context across complex productions, briefing toward a specific output and evaluating what comes back, are the same skills that make agentic AI workflows function rather than fail. The gap between where producers are now and where they need to be is a translation gap. The discipline is already formed. What is missing is the vocabulary in which to apply it.

Engineers Constrain. Producers Brief.

When an agentic AI system produces bad output, two different professional instincts respond to it differently.

The engineer's instinct is to add a constraint. The agent did something it should not have done: add a guardrail, a hard limit, an instruction that prevents the behaviour from recurring. The constraint is applied to the system. It becomes part of the architecture, a negative rule encoded to prevent a specific failure.

The producer's instinct is to fix the brief. The agent did something it should not have done: go back to what you asked for, identify what it did not understand, and reformulate. The fix is applied to the communication. It becomes a more precise description of what you actually need.

Both responses solve the immediate problem. Over time, they produce different systems. The constraint-based approach accumulates layers: rules on top of rules, exceptions to exceptions, a growing catalogue of things the agent cannot do because at some point it did them badly. The system gets more rigid with each correction. The brief-based approach accumulates legibility: a brief refined through iteration becomes more predictable and more useful than the brief it started as. The system gets more reliable.

A 2026 academic analysis of the Claude Code architecture makes this visible in numbers. The researchers found the ratio of AI decision logic to deterministic infrastructure to be approximately 1.6 to 98.4. One point six per cent of the system is AI making decisions. Ninety-eight point four per cent is the operational environment: the harness, the context, the scaffolding, the structured framework within which that intelligence operates. The phrase they used for the design philosophy is "minimal scaffolding, maximal operational harness." [EN: Liu, Jiacheng, et al., VILA Lab / MBZUAI, arXiv 2604.14228, April 2026]

The harness is the producer's domain. The producer who spends time clarifying what the system is supposed to do, what standards its output must meet, what constraints bound it, and what happens

when those standards are not met is building the harness. That work is not peripheral to the AI capability. It is the environment in which the AI capability operates.

The distinction between conditions and constraints makes this concrete. A constraint says what the system cannot do. A condition describes what success looks like: the output must achieve this objective, take this form, stay within this scope, and be evaluated against these criteria. A system that knows what success looks like fails in new, legible ways and recovers. A system defined by its constraints fails in the gaps between them and produces nothing useful in those gaps.

I use the term vibe-production to describe the specific practice of investing in brief clarity rather than constraint accumulation. [EN: Sandberg, "Engineers Constrain. Producers Brief.", Game Production Alchemist, 2025] The name is deliberately casual. The practice is not. Writing a brief that produces what you want from an agentic system requires the same craft as writing a brief for a contractor: specificity about the objective, clarity about scope, an explicit statement of the criteria by which you will judge the output, and an honest answer to what you will do if the output does not meet them. Vibe-production is the name for treating this as a craft discipline rather than a nuisance tax on getting the real work done.

Producers already treat briefing as a craft. The translation is immediate.

The Five Skills

There are five producer skills that transfer to agentic AI workflows without modification. These are not new competencies to acquire. They are existing competencies applied in a new context. Each one appears throughout the rest of this book, named and applied to different problems in different chapters. The names are vocabulary.

Prompt precision is brief writing applied to AI instruction. The skill is scoping an instruction tightly enough that the output is recognisably what you intended, and clearly enough that the collaborator can do the work without being micromanaged. In production, this is the difference between a milestone spec that gives the team direction and one that buries them in detail. In AI, it is the difference between a prompt that produces what you need and one that produces something adjacent and plausible that you have to redo, with no obvious reason why.

The specific challenge with AI is that there are no contextual defaults to lean on. A human collaborator who has worked in your studio for two years brings assumptions about how you work, what you care about, what shortcuts are acceptable. An AI brings none of those unless you put them in the brief. The brief must carry what the human would infer. Most briefs do not, and most of the output problems that producers attribute to AI limitations are actually omissions in the brief.

Output taste is the capacity to evaluate what comes back. In production, this is reading a build and knowing it is shipping, or reading a first draft and knowing which quarter of it is worth keeping. In AI, it is reading generated output and knowing whether it is

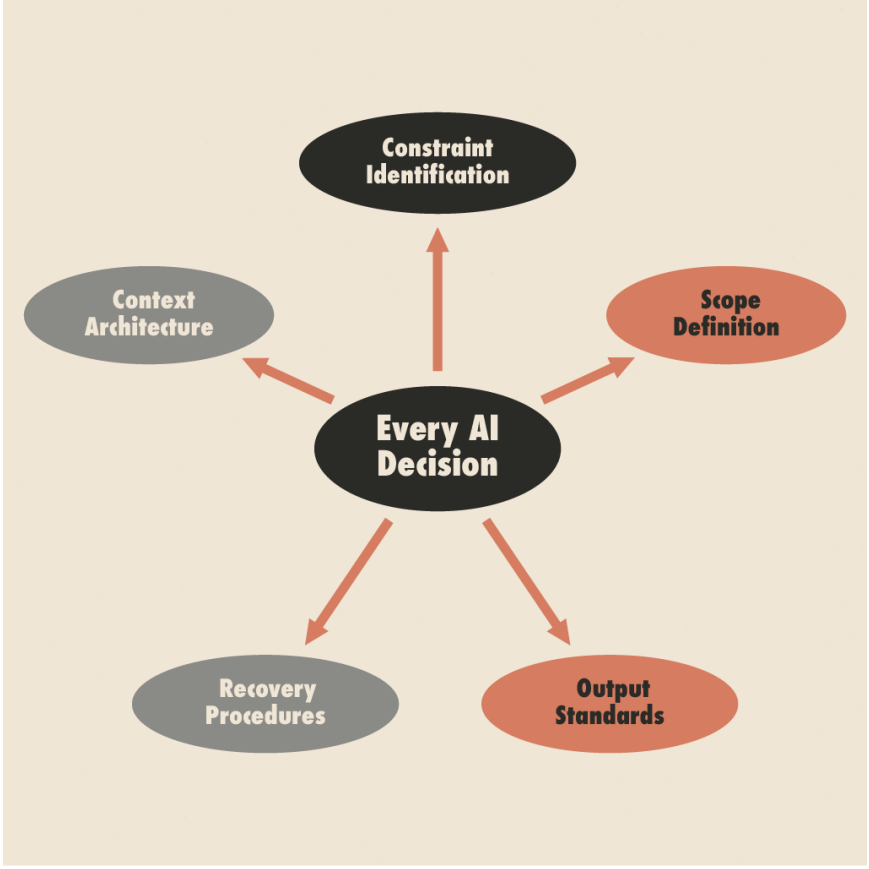
accurate, appropriately scoped, and safe to build on, or whether it is confidently wrong in a way that sounds right. AI does not know what it does not know. It does not flag gaps in its own reasoning or signal where its confidence is unearned. Taste is the producer-side check that the AI cannot provide for itself.

Context architecture is deliberate management of what the system knows. In production, this is the awareness of what lives in the milestone document, what is in people's heads, what the team actually understands versus what it thinks it understands. In AI, it is the awareness of what lives in the system prompt, what is in the current conversation window, what must be re-established at the start of every session, and what is decaying silently as the conversation grows longer. Context in AI is not persistent by default. It is a structure the producer builds and maintains deliberately, rather than one that accumulates naturally through shared work. The producer who already manages information architecture across a production team already has this skill. The domain is new; the discipline is not.

Iteration without attachment is the capacity to treat every output as a draft. In production, the producer who has revised a feature specification seventeen times before it is right has already internalised what this requires: the output is a working document, not a verdict. In AI, the same applies. Refining and redirecting are the normal loop, not signs that the tool has failed or that the task is beyond it. The producer who experiences each revision as a new failure is not fluent in what she is working with. The producer who experiences each revision as a step toward a usable output already understands the collaboration.

Knowing when to take the wheel is the recognition that the tool has reached its ceiling and a human decision is required. In production, this is the producer who knows the team cannot deliver a feature in the time the plan says, and adjusts the plan rather than the team. The situation calls for a decision, and the producer makes it. In AI, the same move applies: when the output is consistently missing what matters, when the problem requires specific judgement that the tool does not carry, when the stakes are high enough that iteration is not an appropriate response: the producer steps in, makes the call, and stops expecting the tool to do what only a person can do.

These five skills form a starting vocabulary for what follows in this book. The remaining chapters apply them to specific problems in specific domains. The vocabulary holds throughout.



The Idiot Savant

There is an analogy that comes up when experienced AI users try to describe what working with a capable agentic system actually feels like. The phrase is idiot savant: a collaborator with a remarkable, focused capability and a complete blind spot in an adjacent direction.

The capability ceiling is surprisingly high. The floor is also surprisingly low. Leave an assumption unspoken and the agent will confidently produce something plausible and wrong. Give it ambiguous direction and it will pick an interpretation and run with it without flagging the ambiguity. It will not push back on a poorly-formed request, will not ask for clarification unless prompted, and will not notice that what you asked for is not what you need. These are not random failures. They are consistent features of this class of collaborator. [EN: Sandberg, "Game Producers Are Built for Agentic Workflows", Game Production Alchemist, 2025]

The management posture that works: be specific, assume nothing, check the work. A well-formed brief for an agentic system contains four elements: a clear statement of the objective, the specific form the output must take, the scope boundaries (including what is out of scope and why), and the criteria by which the output will be evaluated. Without any of these, the system fills the gap from its defaults rather than your requirements. Writing a brief that works with an AI is brief writing with the implicit scaffolding removed: the scaffolding a human collaborator fills in from shared context.

The brief-writing skill transfers further than most producers expect. Codebase legibility is one example. Before agentic AI, a non-technical producer could not meaningfully interrogate a codebase: questions like "how many places does this system touch?" required technical fluency to formulate, let alone to answer. What AI provides is not that fluency. It provides a collaborator capable of receiving a well-formed objective and returning something the producer can evaluate. Prompt

precision, the first of the five skills, is what makes that access possible. The producer's ability to identify the right question and specify it clearly determines the utility of the answer.

My own practice makes the shape of this visible. I am not a technical producer by training. I have never written a function. Working with Claude Code, I built a content processing pipeline and a game prototype, starting from descriptions of problems I understood well enough to articulate. The interface I used was screenshots and plain language. The protocol was "explain step one like I am ten years old." This is a power move rather than a confession of ignorance: it forces a level of specificity that bypasses assumptions.

What I needed to understand was the shape of the thing, not the syntax. Prompt precision got me to the right ask. Output taste, the second skill, told me whether what came back was accurate, appropriately scoped, and safe to build on. When the gap between the two was too large, the redirect was production. [EN: Sandberg, "I Don't Know What I'm Doing. That's the Point.", Game Production Alchemist, 2025] The collaborators were different. The discipline was the same.

What You Bring to It

Producer cognition is architecture, not soft skill. The capacity to brief a collaborator clearly, evaluate output against criteria, manage the context a team operates within, iterate without treating each revision

as failure, and recognise when the decision requires a human: these are the structural skills of production governance. They have always been structural. What changed is the domain in which they apply.

The Claude Code Game Studios repository, the same open-source project referenced in Chapter 1, is publicly visible and its issue threads are publicly searchable. The most-discussed thread across the project's community is a variant of the same question: why won't the AI just make the decision? The architecture's answer is explicit. It is not supposed to. The system is designed to surface trade-offs, price conflicts, and hold the decision point open until a human closes it. [EN: Claude Code Game Studios, GitHub community threads, 2026] Users who expect the system to function autonomously are frustrated by this. Users who treat the system like a production that needs a managing director are not.

This is the translation gap made visible in real time. The question in the thread is not a product complaint. It is an expression of the same category error that produces the HBR piece about the new role: the assumption that a sufficiently capable AI should be able to close its own loop. The agentic systems built by the engineers who understand this best are specifically designed not to. The governance mechanism they arrived at requires a human to stay in the loop at every meaningful decision point. That requirement is the job. Producers already know how to do it.

The gap to close is a translation gap. The mental model is already formed. A career of production discipline: briefing capable collaborators, managing context across complex work, evaluating output against criteria that the system cannot generate for itself, and

making the calls that only a human can make. This is exactly the mental model that agentic AI requires. Chapter 3 builds the system that puts the translation to work.

Endnotes

1. Srinivasan, Suraj, and Wei, Vivienne, "To Thrive in the AI Era, Companies Need Agent Managers," *Harvard Business Review*, February 12, 2026. hbr.org/2026/02/to-thrive-in-the-ai-era-companies-need-agent-managers. Referenced in GPA post "HBR Just Invented the Producer." The article described what producers recognise as core production governance: setting context, translating intent, evaluating output, managing iteration.

2. Claude Code Game Studios, open-source agentic coding framework, GitHub, 2025. Q→O→D→D→A protocol (Question, Options, Decision, Draft, Approval) from CLAUDE.md governance file. Producer agent function: "The producer does not resolve conflicts. It prices them." See Chapter 1, EN5 for full attribution note on star count and initial release.

3. Liu, Jiacheng, et al., "Dive into Claude Code: The Design Space of Today's and Future AI Agent Systems," VILA Lab, Mohamed bin Zayed University of Artificial Intelligence, arXiv 2604.14228, April 14, 2026. Analysis of Claude Code's publicly available TypeScript source: 1.6% AI decision logic, 98.4% deterministic infrastructure. Phrase "minimal scaffolding, maximal operational harness" appears in Table 1 (Section 2.2) as one of thirteen design

principles. Third-party academic analysis, not official Anthropic documentation. Referenced in GPA post "Engineers Constrain. Producers Brief."

4. Sandberg, Rob, "Engineers Constrain. Producers Brief. (Or: what your briefing practice is worth in the age of agentic AI)," Game Production Alchemist (Substack), 2025. Conditions-over-constraints framework; vibe-production as specific brief-investment discipline.

5. Sandberg, Rob, "Game Producers Are Built for Agentic Workflows," Game Production Alchemist (Substack), 2025. Five transferable producer skills named and grounded; idiot-savant framing of agentic collaboration.

6. Sandberg, Rob, "I Don't Know What I'm Doing. That's the Point.," Game Production Alchemist (Substack), 2025. Personal account of non-technical producer using Claude Code. Screenshot-as-interface, plain-language protocol, and codebase legibility as practical translation.

7. Claude Code Game Studios, GitHub issue threads and community discussion, 2026. "Why won't the AI just make the decision?" as a recurring user complaint against the governance architecture's deliberate human-in-the-loop design.

Get the Full Book

Outlast the Tools continues across twelve chapters covering constraint theory, forecasting, flow metrics, codebase legibility, and how to read AI output with the same investigative habit that experienced producers apply to status updates.

Available now at leanpub.com/outlast-the-tools