Biome engineering

Chris von Csefalvay

Blog post published October 11, 2025 in *Chris von Csefalvay*
https://doi.org/10.59350/6yppb-ssv49

Abstract

Roughly 541 million years ago, something extraordinary happened in Earth's oceans. Over a geologically brief period of perhaps 20 million years, the fossil record explodes with an almost obscene diversity of body plans.

Keywords

Epidemiology (Subfield), AI, Agentic AI, LLMs

Recommended citation

von Csefalvay, C. (2025, October 11). Biome engineering. *Chris Von Csefalvay*. https://doi.org/10.59350/6yppb-ssv49

Copyright

Copyright © Chris von Csefalvay 2025. Distributed under the terms of the Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Roughly 541 million years ago, something extraordinary happened in Earth's oceans. Over a geologically brief period of perhaps 20 million years, the fossil record explodes with an almost obscene diversity of body plans. Trilobites, anomalocarids, the bizarre Opabinia with its five eyes and frontal grasping appendage – suddenly, the relatively monotonous microbial world gave way to an abundance of diversification in what we now call the Cambrian explosion. The textbook explanation points to rising oxygen levels in the oceans, changes in ocean chemistry and perhaps the evolution of predation itself. But here's the thing: none of these environmental shifts determined which specific forms would emerge. The changing ocean chemistry didn't decree "there shall be trilobites". Rather, it created conditions that made certain evolutionary pathways more energetically favourable, certain body plans more viable, certain ecological niches newly accessible.

The environment set the stage. Emergence explored it, and wrote the play.

I find myself returning to this image as I think of reconciling two ideas in apparent opposition: the fact that AI agents' true value lies in their ability to independently reorganise and structure themselves in a way that gives rise to an emergence of sorts, with the need to govern such systems and guide them towards the outcomes we desire.

We've moved through distinct phases of how we think about shaping AI behaviour, each building on the last like Schliemann's nine cities of Troy. First came prompt engineering, the art of crafting the perfect utterance to coax desired behaviour from a language model. Then Andrej Karpathy articulated what he termed context engineering – the recognition that an agent's operating envelope, the information and tools it has access to, matters as much as any individual prompt. But as I argued in my recent writing on agent ecosystems, we're now confronting something more profound: the need to engineer not individual agents or even their immediate contexts, but the entire operating environment in which populations of agents interact, specialise and evolve solutions we couldn't have explicitly programmed.

I call this biome engineering. And like the Cambrian ocean chemistry, it's an intricate dance of constraints and gradients that gently shape, not prescribe, a spontaneous emergence.

From utterances to ecosystems

Let me trace the genealogy of this idea for a moment. Prompt engineering emerged from a fundamentally transactional view of AI: you ask a question, the model responds, the interaction terminates. The craft lay in asking the right question in the right way. It was, in essence, the art of the perfect utterance – a rhetorical exercise that would have delighted the Sophists. We developed elaborate techniques: few-shot learning, chain-of-thought prompting, role-playing scenarios. The assumption was that if we could just find the right incantation, we could reliably extract the behaviour we wanted from the model.

This worked reasonably well for a while, but it had a fundamental limitation: it treated each interaction as isolated, acontextual, amnesic. Every conversation started from scratch. Every task required re-establishing context.

https://doi.org/10.59350/6yppb-ssv49

Context engineering represented a conceptual leap. Karpathy's insight was that we should stop thinking about individual prompts and start thinking about the entire informational environment in which an agent operates. What knowledge does it have access to? What tools can it invoke? What constraints govern its actions? More broadly, how can we 'dope' the agent's context with just the right nudging context to create a groove in the gradient space towards what outcome we desired, without explicitly constraining it? An agent with proper context engineering isn't just responding to utterances – it's operating within a defined possibility space, with persistent memory, access to resources and the ability to maintain coherent behaviour across extended interactions. The tradeoff is, of course, that if finding the right utterance was difficult, finding the right context is even more so – and the reward for perfect context engineering is even less deterministic.

But context engineering, for all its power, still focuses on the individual agent. It's about optimising the envelope within which a single agent operates. And this is where we run into the same conceptual wall I've been banging on about in my writing on agent ecosystems: the real value of agents isn't in what they can do alone, but in how they work together.

Leveling up

Here's where we need to make the leap from engineering individual contexts to engineering entire biomes – the substrates in which populations of agents can interact, specialise and collectively solve problems in ways that emerge from the system rather than being explicitly programmed.

A biome, in the ecological sense, isn't just a collection of organisms. It's the entire system: the organisms, their interactions, the physical environment, the flows of energy and nutrients, the constraints and opportunities that shape what can survive and thrive. The Serengeti isn't just lions and gazelles and acacia trees – it's the pattern of rainfall that determines where grass grows, the soil chemistry that shapes which plants can take root, the seasonal migrations that move energy across the landscape.

Biome engineering for AI systems means designing the operating environment for populations of agents: not just what each individual agent can do, but how agents can discover each other, how they establish trust and negotiate collaboration, what resources they can access and under what constraints, how authority and delegation flow through the system. It's about creating the substrate conditions that allow certain interaction patterns to flourish whilst making others energetically unfavourable or outright impossible.

The crucial insight is that biome engineering is non-deterministic in its outcomes but bounded in its possibilities. Just as the Cambrian ocean chemistry didn't produce a specific catalogue of species but rather made certain evolutionary pathways more likely, biome engineering doesn't specify exactly how agents will collaborate – it creates conditions that favour productive patterns whilst constraining destructive ones.

More importantly, this allows for agents to dynamically recombine within a bounded space of possibilities. This makes the collective of agents responsive to adapt to changing conditions, i.e. to

optimise the graph of interactions $G = (V, E)$ so that its performance minimises a loss function over time. And if this sounds awfully similar to the way we train every neural network, that's because it is. Biome engineering is awfully important because it essentially supersedes the legacy approach of deterministically layering a small number of agents. That may work for the travel-planning-in-five-agents toy example, but not for emergent systems. Just as we wouldn't have working neural networks if we had to manually run backpropagation, we need an effective way to let agents self-organise and learn towards a goal at scale. Biome engineering turns that problem into a tractable one.[1]

[1] And, arguably, differentiable. Which is where the real magic comes in, since whatever is differentiable can be computationally optimised using gradient descent. But that's a post for another day.

The biome as substrate

There's a concept in exercise physiology called "training stress" – not stress in the colloquial sense of anxiety, but the physiological stress that drives adaptation. You don't tell your body to build more mitochondria or increase stroke volume. You create conditions (progressive overload, adequate recovery, proper nutrition) that make those adaptations energetically favourable. The body responds to the environment you create. Do it long enough, and you might as well end up doing things you probably didn't think was possible.

Biome engineering works similarly. You're not explicitly programming agent behaviours – you're shaping the problem space they explore by setting its boundaries and its internal gradients. This means thinking carefully about a lot of things we generally don't (or at least not in these terms):

- Resource topology: What information, tools and compute resources are available? How are they distributed? What are the costs (in tokens, time, money) of accessing them? Just as species diversity in ecology is partly shaped by resource patchiness, agent specialisation will emerge based on how resources are structured.
- Interaction protocols: How can agents discover each other? How do they establish capabilities and negotiate terms? What standards govern their communication? This is a problem I've written about – we need something richer than current constructs (I like MCP, it's just limited to what it has been designed to accomplish), something that can convey not just data but trust, authority and constraint.
- Constraint boundaries: What are agents allowed to do, and crucially, how do those constraints propagate through delegation chains? An agent commissioned by another agent must inherit appropriate constraints from its principal. The biome's "laws of physics" need to make certain behaviours impossible, not just discouraged.
- Feedback mechanisms: How do agents learn what works? Do successful collaboration patterns get reinforced? Do agents that consistently meet their commitments earn reputation that makes future collaboration easier? The biome needs something analogous

to ecological fitness – ways for productive patterns to flourish. *Diversity pressures: Monocultures are fragile, whether in agriculture or in agent systems. The biome should create niches that favour specialisation over generalisation, depth over breadth in specific domains.

The art of biome engineering lies in setting these parameters such that the agents' exploration of the problem space naturally tends toward solutions that are acceptable to you, while avoiding failure modes that aren't. You're creating a landscape where certain paths are easier to traverse, certain peaks easier to reach, without explicitly commanding "go climb this mountain".

The non-problem of emergence

I can hear the objection already: "But if you're not explicitly programming the behaviour, how do you know what you'll get?" This is a legitimate concern, particularly in regulated industries where I spend most of my time. The enterprise software world has been built on determinism, on the ability to specify and verify exactly what a system will do.

Biome engineering does require a different relationship with emergence and uncertainty. You're creating conditions, not commanding outcomes. But this isn't as radical a departure as it might seem. We already accept this in other domains. When you build a market, you don't specify every transaction: you create rules, mechanisms for price discovery, constraints on behaviour, and let trading patterns emerge. When you design a city, you don't dictate every social interaction: you create infrastructure, zoning, public spaces, and let communities form. There is an awful lot of room between fully deterministic prescriptive spaces and pants-on-head anarchy.

The key is that whilst specific outcomes aren't determined, the possibility space is bounded. The Cambrian explosion produced wild diversity, but it didn't produce physically or biologically nonsensical outcomes (or even anything that deviates very significantly from the usual order of low level biological functioning) – the environmental constraints and existing biological toolkit limited what could emerge. Similarly, well-designed agent biomes channel emergence within acceptable boundaries.

This requires new forms of verification and validation. Instead of testing whether a system produces specific outputs for specific inputs, you're testing whether the biome's constraints hold under stress, whether emergent behaviour stays within acceptable bounds, whether the feedback mechanisms actually reinforce productive patterns. It's closer to stress-testing a bridge by driving over it a few hundred times than debugging a program.

The orchard and the grove

There's a spectrum here, and different applications will sit at different points along it. Some biomes will be heavily cultivated orchards, with tight constraints and limited opportunities for emergence. Others will be closer to managed wildernesses, with looser boundaries and more room for unexpected behaviour.

High-stakes, highly regulated domains – medical diagnosis, financial trading, safety critical systems – will tend toward the orchard end. You want strong constraints, limited emergence, predictable patterns. But even here, biome engineering offers advantages over purely programmatic approaches. A well-designed biome can enforce regulatory compliance more flexibly than hard-coded rules, can adapt to novel situations within constraints, can allow specialised agents to collaborate whilst maintaining audit trails and accountability.

More exploratory domains – research, creative work, open-ended problem-solving – can afford to sit further toward the wilderness end. Here you want emergence, want agents to discover novel collaboration patterns, want the system to surprise you with solutions you wouldn't have thought to program.

The crucial capability is being able to tune this dial based on context. The same underlying biome infrastructure should support different constraint regimes for different use cases. This is where biome engineering diverges most sharply from both prompt and context engineering: it's not about individual interactions or individual agents, but about the entire operating environment and its relationship to the problem space.

Towards crafting biomes

I'll be honest: we're still in the very early stages of this. Most of what I'm describing doesn't exist in production systems yet. We have fragments – agent marketplaces that enable discovery, delegation frameworks that propagate constraints, trust scoring systems that track reputation. Most of these are nascent at best. But we don't yet have coherent biome engineering frameworks that bring these pieces together.

The technical challenges are considerable. We need standards for agent capability description, protocols for trust establishment, mechanisms for constraint propagation, frameworks for reputation and verification. We need to solve the meta-problem of how biomes themselves can be specified, deployed and validated.

The conceptual challenge is equally significant. Biome engineering requires a different mindset from traditional software development. You're not building a machine with specified behaviour, you're cultivating an ecosystem with bounded emergence. This demands comfort with uncertainty, skill in thinking about system-level properties rather than individual components, and the ability to reason about constraints and incentives rather than explicit commands. It's a shift from architect to gardener, from engineer to evolutionary theorist. And much like the Cambrian explosion, I suspect we're going to see a period of wild experimentation, of diverse approaches and unexpected solutions, before things settle into more stable patterns.

But then again, perhaps stability isn't the goal. Perhaps the point of biome engineering is to create substrates where continuous adaptation and evolution are not bugs but features – where the system remains dynamic, responsive and capable of surprising us long after we've deployed it. There's no reason why this needs to be the *only* model – deterministic, programmatic systems will still have their place, just as LLMs didn't displace the template engines that deterministically

https://doi.org/10.59350/6yppb-ssv49

generate the bulk of the web. But for complex, open-ended problems, biome engineering offers a promising path forward.

The dramatic perturbations 541 million years ago that gave rise to an abundance of weird attempts at existence, of which some proved successful enough to thrive, continue to ripple into the present. Evolution has kept improvising new variations on those original themes that allowed life to persist and thrive despite drastic changes to the face of the earth. If we do biome engineering right, our agent ecosystems might show similar staying power and potential.

https://doi.org/10.59350/6yppb-ssv49