A READING MAP

Deleuze on Problems

— and what it means for building software

> *"A problem always has the solution it deserves."*
>
> GILLES DELEUZE · BERGSONISM

How a problem is constructed already shapes the solution that extends from it.

I · BERGSONISM — THE ROOT

A problem has the solution it deserves

Truth and falsity belong first to problems, not to answers. Most error is a competent answer to a badly-posed question. Posing a problem is invention, not discovery: state it rightly and its solution already exists, waiting only to be uncovered.

A problem can go false in two ways. Its terms may smuggle in a confusion — "why is there something rather than nothing?" quietly assumes that nothing is *less* than something, when the idea of nothing is in fact *more*: it adds, to the idea of something, the mental act of abolishing it. Or a composite may blur a difference in *kind* into a mere difference of *degree*.

Bergson's method has three rules: apply the test of true and false to problems themselves; dispel false differences of degree to recover the real differences in kind; and state problems in terms of time rather than space. Its central movement is to cut the composite along its real articulations, then follow the divergent lines back to the virtual point where the problem is properly posed.

> THE ENGINEERING TRANSLATION
>
> **Problems get the solution they deserve.**
>
> The architecture of the problem fixes the solution space. A well-posed problem is already solved in principle — the rest is typing. Most bad code is a clean solution to a badly-decomposed problem. Leverage is in framing, not cleverness: the bug that evaporates once you state the requirement correctly is a false problem dissolving.

II · DIFFERENCE AND REPETITION — THE ONTOLOGY

The problem as Idea — relations and singular points

The problem becomes fully real as an *Idea*: a structured multiplicity of differential relations and singular points. Its elements have no identity on their own — like `dx` and `dy`, each undetermined in isolation, they determine one another entirely through their relation. The singular points are the turning-points where behaviour changes, and their distribution *is* the problem.

Here is the load-bearing distinction. **Differentiation** — with a **t** — names the problem's virtual structure, its differential relations and singularities. **Differenciation** — with a **c** — names its actualization. Integration extends the differential, but the solution does not pre-exist in the problem: it is engendered by divergence, not unrolled from a finished form.

The problem does not vanish when it is solved. It persists, and insists, in its solutions. To learn is to rise to the level of the problem; knowledge is only the stock of solutions.

THE ENGINEERING TRANSLATION

Differential relations = interfaces.

Components are defined by their relations — contracts and API boundaries — not their internals. Like `dx` and `dy`: each is undetermined alone; it is their relation that is completely determined.

THE ENGINEERING TRANSLATION

Singularities = the seams.

Edge cases, boundary conditions, phase transitions, interfaces — the points where behaviour changes. Mapping a problem means locating its singular points, which is what factoring a system well already is.

THE ENGINEERING TRANSLATION

differentiation → differenciation.

The virtual is the problem itself — the tension immanent to the domain. Spec and build are successive actualizations of it at different grains; the spec is already a partial determination, not the virtual. Actualizations diverge and need not resemble the problem — which is why two sound architectures for the same need can look nothing alike.

III · THE LOGIC OF SENSE — THE PROBLEM ON THE SURFACE

Singularities, series, and sense

The same structure returns, recast as *sense* and *event*. Singularities — inflections, knots, bottlenecks, pre-individual and "nomadic" — are distributed over series

that communicate through a circulating paradoxical element. Sense is produced at the frontier where two series touch. The event, like the problem, persists in its solutions without being used up.

> THE ENGINEERING TRANSLATION
>
> **Sense lives at the seam.**
>
> Meaning — and most defects — appear at the frontier where two series meet: the boundary between two subsystems, the protocol where they resonate. Code the seams, not just the parts.

IV · DURATION & BECOMING — WHY IT MUST BE THE VIRTUAL

The problem is virtual, not possible

Why insist on duration and becoming? Because the problem is *virtual*, not *possible*. The possible is pre-made and merely "realized" by resemblance and limitation — the real only adds existence to a form already complete. The virtual is fully real and actualized by genuine creation: the solution invents, it does not copy. The virtual problem is like the coexisting past — drawn on in every present without being exhausted. So the problem stays a verb — an ongoing differenciation — never a finished noun.

THE ENGINEERING TRANSLATION

Virtual vs. possible (Claude / Codex / Copilot).

It is tempting to say a code model actualizes a solution that did not pre-exist — but keep it an analogy, not an identity, because it cuts both ways. A model's weights are a vast compressed store of patterns, and sampling selected by fit to the prompt looks a lot like Bergson's possible *realized by resemblance* — the combinatorial pre-given — rather than a virtual that produces what does not resemble its condition. Whether model output is actualization or sophisticated interpolation is exactly the open question; keeping the objection visible is the more Deleuzian move.

THE ENGINEERING TRANSLATION

The problem persists in the solution.

It does not vanish at ship. The codebase keeps the shape of the problem it answered, and the problem re-actualizes through versions and refactors. Software is becoming, not being — "done" is always provisional.

FIVE ONE-LINERS FOR THE CONVERSATION

— Spend your time on the problem, not the answer.

— Find differences in kind, not degree.

— Code at the singularities.

— The problem is virtual; spec and build are both actualizations.

— A model may interpolate the possible, not actualize the virtual — ask which.

REFERENCES

Bergson — *Bergsonism*, ch. I "Intuition as Method" `(§BERG-01)`; the virtual / possible distinction and the coexisting past in ch. III "Memory as Virtual Coexistence" `(§BERG-03)` and ch. V "Élan Vital as Movement of Differentiation" `(§BERG-05)`.

Difference and Repetition — ch. IV "Ideas and the Synthesis of Difference" `(§DR-04)`; the critique of the dogmatic image of thought, ch. III "The Image of Thought" `(§DR-03)`.

The Logic of Sense — Ninth Series, "Of the Problematic" `(§LS-S09)`.