Dimensional Design

1 The Dimensional Gap

A computer takes in everything as one long line of zeros and ones. That is all it gets: a single stream, one thing after another. A picture, a song, an equation must all be built from that flat line. Computers are intrinsically one-dimensional.

People are not like that. We spent millions of years evolving to live in a world with many dimensions, not just the four we can measure (width, height, depth and time) but far more that no ruler captures: colour, smell, hierarchy, trust and even emotions. People take the world in across all of them at once. People are intrinsically multi-dimensional.

Between multi-dimensional people and one-dimensional computers sit computer programs.

We are used to thinking of computer programs as clockwork machines. A clockwork machine follows instructions. Give it the same instructions and it gives the same result, exactly the same result, every single time. There is no guessing inside a clockwork machine.

We have a new kind of computer program that is not a clockwork machine. It is the thing most people call AI. It works by guessing what comes next from the huge amount it has seen before. It is a predictive machine. Its output is *probably right*, not *exactly right*.

2 Compounding Probably Right

The real world does not run like clockwork, and neither do we. We deal in *probably right*. Traditionally, every computer program demanded the careful precision of a clockmaker; a predictive machine is the first that does not, and that is why it feels so different to work with. Underneath, it is still a computer program, still reading along a single dimension.

When a predictive machine is asked to build something with lots of hidden dimensions, lots of all-or-nothing pieces chained together, it can get dangerous. *Exactly right* chained to *exactly right* stays *exactly right*, every time; *probably right* chained to *probably right* becomes almost certainly wrong, eventually.

We teach our children that chaining *probably right* eventually leads to almost certain error through the game of telephone. A message is passed around a circle of children. Each child's understanding of the message they received is *probably right*, but what comes out when it gets back to the start is almost certainly wrong. Each retelling adds a small doubt and the doubts compound. What is new is seeing it in computer programs. Until recently, computer programs have been clockwork machines: chaining one after another has always been safe, because each step gave back the same result every time. Predictive machines break that assumption, and our mental models have not caught up.

3 Different Routes to the Answer

How does one check a stack of figures from paper invoices entered into a spreadsheet for errors? Repeating the task and investigating the discrepancies works only if the two attempts fail independently. A smudge that is always read the same wrong way defeats it. Both passes misread it identically, the two attempts agree, and the error hides. Summing the digitized values and comparing the result against the printed total on the invoice catches it instead. That total was produced separately from the typing, so a single misread digit throws the sum out of agreement with it. The check tests the work on a different dimension from the one the typing used, and it would be a wild coincidence for two separate mistakes to cancel each other out exactly.

A check on an independent dimension is what is missing from how we usually check a predictive machine. Too often a reviewer is handed the output and asked to read it line by line for errors. When the predictive machine is right ninety-nine times out of a hundred, catching the one wrong case that way is a task no one can do reliably. The failure is in the task, not the reviewer, and that is why a different route is needed.

Running the predictive machine many times and taking the majority does not help either: repetition buys agreement, not correctness. To know whether the answer is right still requires checking it on a dimension that does not share its blind spot. A check built that way reports only that an error exists somewhere, not where it is, and does so without re-reading the work piece by piece.

4 Plain Text for All

A file, at the level the computer actually reads it, is a long line of zeros and ones. That is what the computer was built to read. A plain text file adds almost nothing to that stream: each grouping of eight stands for a letter or a space or a line break, and the words sit in the order they were written. The file is its own description; a person can read it the same way the computer does.

A fancy document (Word, PowerPoint, PDF) is the opposite. It buries the words inside a thick schema of rules about fonts, spacing, layout, headings and a great deal else. Each rule is a hidden dimension the file silently carries. To read the file, a program first has to decode every dimension of the schema; only then does it know which parts are the writing.

The cost shows up when a predictive machine is asked to help edit a fancy file. A clockwork machine can hold every hidden dimension *exactly right*, every time. A predictive machine working in *probably right* has to hold all of them right at once, with the small doubts at each step compounding into almost certain error. Plain text avoids the trap: there are no hidden dimensions to keep coherent. The file lives in the one dimension the computer was always reading along, which a person can read directly too, so both can work on it together.

Any fancy layout can come at the very end, applied once by a clockwork machine that does the same thing every time. Until that point, the content stays in the form both person and machine can read.

5 Design the Checks

Some checks offer genuine validation. Other checks are merely performative. What separates them is design: a genuine validation is built so that errors cannot slip through; a performative check assigns blame, but catches nothing.

In double-entry accounting, every transaction between accounts is recorded twice, once as a debit on one account and once as a credit on another. The books balance only when total debits equal total credits. For centuries this is how the books of businesses could be trusted, not because anyone had read every transaction but because an error on one side throws the books out of balance; a mistake would have to be matched, exactly, to escape notice. The trust was never in the bookkeeper; it was in a design that leaves an honest mistake nowhere to hide.

Most checks are not designed; they have been inherited. Predictive machines make the problem hard to ignore. They produce more output, faster, than any inherited process was built to absorb, and what they produce can read plausibly even when it is wrong.

A genuine validation on a predictive machine is a clockwork machine that fails when the wrong answer is produced; the predictive machine then tries again, and again, until the test passes.

This is not the empty repetition of the majority vote; each attempt meets an independent check that fails on a wrong answer. The validation makes the answer as right as the check is well designed.

Where no clockwork machine can exist, the genuine validation must be a task small enough that the reviewer can do it, recorded so each item gets individual attention.

6 Why This Matters

Pascal, Churchill and Lincoln all observed that a short letter takes longer to write than a long one. It uses less ink and less paper and looks like less work, yet getting there means cutting everything that does not need to be said, and that is the slow part. The hard part was never the writing. It was saying exactly what was meant and nothing more. As predictive machines make the work itself faster and cheaper, that is the part left to us: knowing exactly what we want, and caring enough to get it right.

Predictive machines can now produce, in seconds, the output of work that used to take decades of training. The value of that training was never only the skill at the end of it. It was the judgement built up along the way: knowing when to be careful, when *probably right* would do and when *exactly right* was required.

As more and more is done by predictive machines, the gap between work done carefully and work done carelessly will only get bigger. Dimensional Design is the careful way of working across that gap: keep the *probably right* is good enough jobs apart from the *exactly right* is needed jobs, and use the right machine for each.

The Manifesto for Dimensional Design

Predictive machines feel close to us because they deal in *probably right*, but underneath they are still computer programs, still one-dimensional, while people live in many dimensions. The gap has not closed; it has only become harder to see. We are finding careful ways of designing around the gap and helping others do so too. Through this work and in homage to the Manifesto for Agile Software Development, we have come to value:

- **Approximation where it is safe** over **precision where it is not needed**
- **Checks on an independent dimension** over **re-reading the same work**
- **Content that explains itself** over **structure hidden behind a schema**
- **Genuine validation** over **the performance of oversight**

That is, while there is value in the items on the right, we value the items on the left more.

The principles below are the same idea broken into the moves it takes:

1. Know the difference between when *probably right* is good enough and *exactly right* is needed, and name it before taking action.
2. Keep the predictive machine in the space where approximation is safe; cheap to run, safe to repeat.
3. Where a hard boundary must hold, enforce it with a clockwork machine, and let the predictive machine try again until it holds.
4. Where no check can be enforced by a clockwork machine, design the review so the check is genuine, not merely performative.
5. Validate on an independent dimension from the one worked on; a second look along the same dimension often proves almost nothing.
6. Keep content in its simplest, lowest-dimensional form, and add dimension (formatting, layout, structure) only at the point of publication, the same way every time.
7. Predictive machines bring a new category of error; it must be caught and contained by design, not by luck.

Predictive machines have made *probably right* cheap. *Exactly right* remains the part left to us.

