SINCE 1994

Towards *true* engineering.

MOVING SOFTWARE BEYOND CRAFT TO PROFESSION · REX BLACK, INC.

Would you cross a bridge

built by craftsmen?

Crafts are perfectly fine for **dressers, swords, and stained glass.**

Bridges, aircraft, pacemakers, power grids — and the autonomous systems that deliver all of the above — **are not crafts.**

WHAT THIS IS ABOUT

In plain English.

Software is still practiced more like a *craft* than an engineering discipline. It relies on skill, apprenticeship, and process. It is short on materials with known properties, mathematics that predicts behavior, modeling that stress-tests designs, and meaningful standards.

This talk argues:

- **What engineering actually means** — and how software falls short.
- **What the trajectory looks like** — mechanical, civil, aeronautical analogies.
- **What levers move software forward** — materials, math, modeling, standards,

DEFINE ENGINEERING

Science · math · technology.

Engineering is the application of science, mathematics, and technology to the construction of useful things.

Software has the **technology.** It comes up short on the **science and math** — no mature predictive theory of why our systems behave as they do.

In the absence of that theory, we lean on **skill and process.**
That lean is the signature of a **craft.**

ANALOGY 1

From guilds to *a profession.*

ARMS, FROM CRAFT TO ENGINEERING

Swordsmiths vs. materials engineers.

CRAFT (THEN)

- Guilds and apprenticeship.
- Master-craftsman status.
- Guild control of entry.

ENGINEERING (NOW)

- Standardized materials (*1086 steel*).
- Mathematical models of stress and fatigue.
- Published properties.

SOFTWARE TODAY

Entry gated by informal reputation and apprenticeship.
Standardized practice varies enormously by shop.
When something goes wrong, the individual practitioner is blamed — **not the process, not the profession.**

Still in the guild era.

ANALOGY 2

Roman aqueducts.

Seoul subway.

EMPIRICAL ENGINEERING

Works until it doesn't.

ROMAN CIVIL ENGINEERING

Aqueducts and arenas that survive today. Accumulated experience of what shapes worked.

No Newtonian physics. No calculus.

Software is here: we build systems that work, but we can't reliably **explain why.**

SEOUL SUBWAY — BEYOND EMPIRICAL

The longest subway system in the world.

Cannot be built empirically. Requires **true engineering backing the design.**

Empirical gets you a CRUD app. It does **not** reliably get you a pacemaker, autonomous vehicle, or power-grid controller **without a series of high-consequence failures along the way.**

ANALOGY 3

Hong Kong's Tsing Ma Bridge.

A wind tunnel.

THE ROLE OF MODELING

Materials + math + calibration.

The challenge: **typhoons.** Stress-testing a completed multi-kilometer bridge is impossible.

Engineers built a **1:400 scale replica** in a wind tunnel. Ran **Monte Carlo simulations** against storm and failure modes.

What made it possible:

- **Materials** with known properties.
- **Mathematics** of the relevant physics.
- **Models** calibrated against reality.

ENGINEERING MATERIALS · THE MISSING PIECE

Published, tested, reusable.

Other disciplines build with materials whose properties **are published and known:** strength, reactivity, melting point, fatigue curve, thermal expansion.

Software is built by combining **words** that get translated into CPU instructions.

We do not yet have software engineering materials.

Open-source libraries and cloud services are the start — but they mostly publish **features**, not **behavior envelopes.** The gap between *"functional documentation"* and *"materials specification"* is where our worst failures live.

FAILURE DRIVES STANDARDIZATION

UK railway bridges.

The UK railway-bridge collapses drove standardization — of materials, processes, and certifications.

Every mature engineering discipline has this inflection point. Enough catastrophic failures to force the profession to stop and codify.

Software has started this. **CMMI, ISTQB, DO-178C, IEC 62304, ISO 26262.** Real progress.

Still significant inconsistency across the industry. Some worry about premature standardization; others argue **even imperfect standards self-correct faster than no**

SELF-REGULATION · LICENSURE · CERTIFICATION

Wyoming. 1907. *A hundred years late.*

US engineers are **self-regulated** like doctors and lawyers. Governments enforce.

The **first US engineering-licensure law** was enacted in **Wyoming, 1907** — roughly a century after the profession existed — because *"anyone could work as an engineer without proof of competency"* was deemed unacceptable for public safety.

Software sits where Wyoming 1907 sat.

ISTQB, CSTE, various vendor specializations are the first serious attempts by the

ANALOGY 4

Kitty Hawk to orbital flight.

80 years.

AERONAUTICAL ENGINEERING

A trajectory *can* be rapid.

- Wright Brothers flew at Kitty Hawk · **1903.**
- Gagarin orbited Earth · **1961.**
- Apollo 9 tested the lunar module · **1969.**
- Space Shuttle's first orbital flight · **1981.**

Under 80 years from first flight to a reusable orbital spacecraft.

The analogy isn't perfect. But the shape is clear: an engineering discipline's trajectory can be **rapid** once the materials, math, modeling, and standards come together.

A true software *engineering profession.*

SIX MARKERS

What we're building toward.

- **Craft approaches relegated** to hobby and personal projects only.
- **Engineering materials** — standard, tested, reusable components with known and published behavior envelopes.
- **Mathematics** that represents software behavior with useful predictive accuracy.
- **Modeling** that's increasingly precise as the math matures.
- **Continuously refined, meaningful standards** — not checkbox compliance.
- **Government recognition** of reputable certifications — in development and testing.

None of these are hypothetical. All of them exist in some form today. The work is to

WHAT ANY ENGINEER CAN DO NOW

The profession transforms *from below.*

It won't transform from the top. It transforms when individual engineers, testers, and test managers insist on a little more engineering in each program they run.

- **Measure** before you standardize.
- **Standardize** before you scale.
- **Write the model down.**
- **Compare** the model to the outcome.
- **Contribute** to the standards you use.

A generational argument.

Make it a this-year practice.

TAKEAWAYS

Four to hold against.

- **Software is still more craft than engineering.** Observation, not attack. Accept the diagnosis before arguing about treatment.
- **Process and skill are compensating for missing theory.** Not wrong. Not the endpoint.
- **Modeling is where testing belongs.** Performance modeling, load modeling, AI evaluators — the early software wind tunnels.
- **Progress is generational, and it has happened before.** Kitty Hawk → orbital in 80 years. The timeline is long, not impossible.

> *Crafts are perfectly good for building dressers and stained-glass windows. Would*

SINCE 1994

Thank you.

REX BLACK, INC. · REXBLACK.COM/RESOURCES/TALKS/TOWARDS-TRUE-ENGINEERING