Continuous Science Foundation

Case Studies

CASE STUDY – CONTINUOUS SOFTWARE DEVELOPMENT

CONTINUOUS SOFTWARE DEVELOPMENT

HOW AGILE AND LEAN PRINCIPLES RESHAPED HOW WE BUILD, COLLABORATE, AND DELIVER

BACKGROUND

In the early days of software, teams followed a rigid, step-by-step "waterfall" model. Projects took years, were delivered late, often missing the mark entirely. Development was siloed. Decision-making was top-down. Changes to the plan were seen as a threat.

In the 1990s and 2000s, software teams began looking for inspiration—notably to Toyota's lean manufacturing system. Toyota had revolutionized production with just-in-time delivery, root-cause analysis, small batch sizes, and respect for workers as problem-solvers. Software teams asked: what if we built software this way?

The result was a new ***mindset***—and with it, a revolution in practices: Agile development, DevOps, continuous integration, and continuous delivery.

The Transformation

Software teams didn't just copy lean manufacturing. They adapted it to fit the radically different world of code. They built continuous systems that worked iteratively, in small, meaningful units, giving teams the ability to release changes at any time, with high confidence—and to keep improving.

Some of the foundational practices of continuous software development include:

- **Working in small batches:** Avoiding delays and getting fast feedback.
- **Continuous integration:** Merging code fast to reduce defects & increase velocity.
- **Testing** is part of development, not an afterthought—enabling safe, rapid changes.
- **Continuous delivery** pipelines so that value can flow directly from idea to end user.
- **Information is visible** across teams: requirements, blockers, progress, & metrics.
- **Empowered teams:** cross functional, talking to users, identifying problems, proposing solutions, and improving the system.

KEY INSIGHTS

Change is not a disruption to be managed. It's a constant to be designed for. Continuous development didn't just speed things up — it fundamentally reframed the goals of the system: from documentation to delivery, from control to trust, from silos to collaboration.

The shift wasn't just technical. It was cultural and organizational. It treated change as constant, valued learning, and **viewed speed and quality not as trade-offs, but as outcomes of a healthy system.**

CASE STUDY – CONTINUOUS SOFTWARE DEVELOPMENT

INSPIRATION FOR SCIENCE: REDESIGNING HOW WE SHARE KNOWLEDGE

Just as Agile reframed how software teams work, continuous and integrated practices in science invites us to rethink how knowledge is produced, validated, and shared. Science today often mimics waterfall: long timelines, disconnected teams in charge of deployment (publishing) vs development (the science!), infrequent releases (publications), and huge rework at the end.

THE CHALLENGE

Today's scientific system assumes that work must be polished before it is shared. But this model creates bottlenecks:

- Months (or years) of work go unseen until publication and even preprints.
- Valuable feedback comes too late to be corrective and instructive.
- Collaboration is limited to narrow, formal channels and is often adversarial.
- Scientists waste time on administrative burden, slow processes, and rigid expectations.
- Reproducibility suffers because the artifacts of the process — code, data, intermediate results — are missing or fragmented.

THE PARALLEL OPPORTUNITY

Agile and continuous methods reshaped software by empowering teams to experiment, release, and improve in short loops. What if science worked more like that? Not copying software, but asking: what are the principles that matter most — and what new practices could support them?

Designing for Continuity in Science Might Include:

- **Sharing research-in-progress** to invite early feedback and foster collaboration.
- **Modularity**: Breaking monolithic papers into smaller, structured, reusable components.
- **Continuous review**: Peer feedback as an ongoing conversation, not a single checkpoint.
- **Versioning**: Treating content and data like code—version-controlled, testable, & reusable.
- **Automating tedious tasks**, so scientists can focus on science.

THE BIG IDEA

Just as Agile transformed software from a fragile, delay-prone endeavor to a fast-moving, quality-centered practice, continuous practices in science could redefine the pace & impact of research—**starting with a change in mindset, reinforced by changes in systems**.

Transformation isn't about doing the same things faster—it is about redesigning the system to be more human, more scalable, and more effective. *More science-y?*

What are the practices of a more responsive, rigorous, and inclusive scientific system?
What mindsets can we reframe with this lens?
How can we look at our systems differently?
What is continuous science?

CSF

continuousfoundation.org