Avoiding digital transformation failure

Enable, accelerate, and enforce desired behaviors with a trusted software supply chain

Build a trusted software supply chain (TSSC)

A TSSC enables, accelerates, and enforces behaviors and practices that promote a DevSecOps culture, such as test-driven development (TDD) and continuous integration and delivery (CI/CD).

With Red Hat® OpenShift® Container Platform as the foundation, a TSSC brings together trusted third-party tools and prescriptive workflows. DevSecOps teams and users can trust that software produced by a TSSC meets your company's standards for security, compliance, privacy, and transparency.

Challenge

Nearly four out of five digital transformation efforts fail.[1] When they do, companies wonder why their technology investments did little to achieve business goals like faster time to market, improved customer experience, and new service models.

When digital transformation fails, it is often because people and processes do not get the same attention as technology. We identified essential elements of successful digital transformation projects: leadership, product, development, architecture, and operations. Weakness in even one of these elements can reduce the value of transformation or derail the effort entirely. Planning investments with these elements in mind helps prevent common failures and produce demonstrable business outcomes (table 1).

Table 1. Elements of a successful digital transformation

Element of transformation	Capability	What failure looks like
Leadership	Actively support new ways of working	Teams are afraid to try something new
Product	Have a strategy to define the right thing (software that users need, want, and can easily use)	Software does not support the need
Development	Build the right thing	Software does not meet defined need requirements
Architecture	Build the thing right	Software is inflexible and does not scale
Operations	Keep the thing running	Ongoing outages and incidents

Companies that build competencies in each of these areas can more quickly adapt to evolving business needs and improve the user experience.

Elements of a successful digital transformation

Leadership

Promote a culture of experimentation and collaboration. Encourage transparent and honest communication and make the shared goal everyone's goal.

facebook.com/redhatinc
@RedHat
linkedin.com/company/red-hat

redhat.com

1 *Everest Group, "78% of Enterprises Fail to Scale and Sustain Their Digital Transformation Initiatives," 6 Aug. 2018.*

Model and reward good behaviors. Attempts to create a new culture through meetings, memos, or retreats will not work without the right incentives. Reward behaviors that support the new culture, such as:

- Creating common purpose and vision.
- Pushing decisions down the chain of command to the people doing the work.
- Breaking down unnecessary barriers between departments.
- Investing in and encouraging continuous learning.
- Fostering a non-blaming culture, where people aren't punished for failure.

Product

Shift the focus from projects to products. Projects are planned upfront, development stops when the plan is completed, and ongoing maintenance might not be funded. In contrast, products evolve continuously in response to changing user needs and market forces. The product plan includes ongoing development and operations.

Treat the product as an experiment. Formulate a hypothesis and then build a prototype to test it. The experiment succeeds when it proves or disproves the hypothesis. Banco Galicia, for example, experimented with personalized recommendations and self-service to achieve faster onboarding. A global hotel group experimented with adding digital room selection and keys to improve guest satisfaction. Product plans rarely remain intact after user testing.

Do not hesitate to discontinue failed experiments. If the product does not do what users want—or is too difficult to use—try a different experiment. The idea is to fail fast, fail often, and learn always.

Development

Develop empathy for other teams and users. Product teams tend to specify excessive details (including implementation), while operations teams often want to slow down to avoid problems. In a culture without trust, this dynamic leads to added work and friction. To build trust, encourage frequent communications between teams so developers understand the consequences of coding decisions. Automate checklists for policy and acceptance criteria. See the sidebar, "Build a trusted software supply chain."

Aim for rapid feedback. Get the minimum viable product (MVP) out quickly and resist the impulse to over-analyze or over-engineer. An MVP delivers value and can immediately be deployed to production. With short feedback loops, product managers and developers know sooner if they are off course and can swiftly make corrections. MVPs also allow teams to demonstrate new capabilities to get funding.

Foster a culture of curiosity and inquiry. Talented developers thrive when given the freedom to solve hard problems creatively. Creating an environment where people can develop their skills builds better talent and helps improve outcomes, recruiting, and retention.

Commit to technical excellence, testing, and continuous integration. When developers are freed from repetitive work, they can focus on innovation and quality. Automate testing and deployment pipelines to improve code quality and throughput. Encourage developer creativity while holding everyone to high standards.

Architecture

Consider trade-offs. Every decision involves balancing cost, performance, and failure characteristics. Make these choices deliberately. For example, incremental performance improvements can be quite costly, so avoid over-engineering for higher performance than users need and expect.

Create an event-driven architecture. With loosely coupled components, services that produce event notifications do not need to know which services are listening for the notifications and the event does not need to know its consequences. Develop skills in microservices techniques such as service mesh, circuit breaker patterns, caching, and service discovery—or work with a skilled partner.

Enforce architecture with standardized services. When the platform enforces standards, developers can focus on domain logic instead of design patterns. Standardized application architectures use standardized instrumentation to report real-time performance metrics and simplify troubleshooting. See the sidebar, "Remove friction from software delivery."

Remove friction from software delivery

- Make the right way the easy way with service automation.
- Authorize the right people to do the right things with policy automation.
- Show people the consequences of actions with short feedback loops.

Operations

Develop empathy for developers and product managers. Rather than saying "no" to projects or timelines that operations cannot support, work closely with development and product teams to come up with acceptable solutions. Aim to deploy new features at the fastest pace that can be managed. See the sidebar, "Start on the path toward site reliability engineering."

Implement mechanisms for fault detection and fault determination. Monitoring is the foundation of reliability. The goal is to improve understanding of the system and respond appropriately to changes.

Develop the technical understanding to remediate and prevent incidents. Operational excellence requires the capabilities and judgment to respond to incidents in a live system while minimizing service disruption. Conduct post-incident analysis with the goal of learning—not blaming.

Start on the path toward site reliability engineering (SRE)

In the SRE model, the operations team and the business give developers free rein to deploy new code—but only until the error budget is exceeded. At that point, development stops and all efforts are redirected to technical debt. SRE is practical only if development and operations function as one team (DevSecOps). This approach is not an option if you outsource either or both functions to third parties.

Enabling the right behaviors with a trusted software supply chain (TSSC)

Knowing the behaviors required for successful transformation is one thing, but making them happen is another. Imagine a development team leader who issues standing end-of-day orders for team members to check in, code, test, send the test report, and deploy code to the shared environment if the test succeeds. But what if a team member checks in and deploys code, skipping the steps in between? Most organizations lack the safety measures to enable, accelerate, and enforce the right behaviors.

A TSSC accelerates recommended behaviors to lead digital transformation efforts and produce positive outcomes. It can automatically enforce policy requirements for testing, vulnerabilities, architecture, and instrumentation. Development and deployment become repeatable and reliable.

Learn more

Red Hat Consulting can work with you to build a TSSC and develop capabilities in each of the elements of a successful digital transformation. Contact Red Hat Consulting to get started.