Product Thinking (Expansion of the SGEP)

From Output to Outcome – Product Management for Real Impact

Ralph Jocham Magdalena Firlit

2026-01-18T09:00:00Z

Collected Resources for Scrum Guide Expansion Pack
This document is a collection of independent works. Each section retains its original license or copyright status, as indicated. Please refer to each section for specific usage rights and requirements.

Introduction

In 2001, seventeen software practitioners gathered in Snowbird, Utah, and wrote what became known as the Agile Manifesto. It was a short declaration of values and principles that guided a generation of software teams [1]. Almost a decade later, in 2010, Kent Beck — one of the original signatories — spoke at the Startup Lessons Learned Conference and reflected on how he might have written the Manifesto differently or how it should have been then [2]. His version placed new emphasis on team vision, validated learning, customer discovery, and initiating change.

He expressed this shift in values as follows:

Team Vision and Discipline over **Individuals and Interactions** over **Processes and Tools**
Validated Learning over **Working Software** over **Comprehensive Documentation**
Customer Discovery over **Customer Collaboration** over **Contract Negotiation**
Initiating Change over **Responding to Change** over **Following a Plan**

Why this change of mind? The world of 2001 was very different from the world of 2010, and even more so from today. In 2001, software was often delivered by consulting companies, called vendors. A business would hire these vendors because it lacked either the capacity or the expertise to build software in-house. The vendor's job was simple: build what the client asked for, deliver it on time, and maintain quality. In that context, success meant output delivered according to specification.

That was in 2010. For reflection purposes only, one might wonder what a comparable emphasis could look like in 2026.

People Empowerment Improvement over **Individuals and Interactions** over **Processes and Tools Value Improvement** over **Working Product** over **Comprehensive Documentation**
Insight Improvement over **Customer Collaboration** over **Contract Negotiation**
Capability Improvement over **Responding to Change** over **Following a Plan**

In a nutshell:

Short- to long-term Work Climate, Learning, Outcomes, and Impact** over **Outputs ***over** **Activities and 'Resources'***

This reflection revealed a deeper truth: **output is not the same as outcome** [17]. Delivering features is not the same as creating value. And in today's continuous innovation economy, where companies live or die based on customer engagement, product adoption, and financial sustainability, it is no longer enough to just deliver a product. Effective product work requires connecting inputs to activities, activities to outputs, outputs to outcomes, and outcomes to impact [3].

This shift is not just about **what** is done, but about **why** and **for whom** it is done [4]. Too often, teams obsess over delivery speed or process optimization while losing sight of the purpose behind their work. The original intent of Agile, when applied with the **why** and **who** in mind, forces us to ask: *How does this activity or output help customers and strengthen the company's future?* That question grounds day-to-day work in long-term impact.

This document explores that shift. Using the lens of the Scrum Guide Expansion Pack (SGEP) [14] and frameworks such as Evidence-Based Management (EBM), this analysis explains why product teams should shift from focusing on activities and outputs to intentionally owning outcomes and impacts [5].

Why the Agile Manifesto Looked Like It Did

To understand why Kent Beck's 2010 version looked so different, the context of 2001 must be revisited. The authors of the Agile Manifesto were a potpourri of mostly consultants, software creators, experts, and thought leaders behind various upcoming practices. They were deeply involved in improving software development processes. Their business model was built on delivering software projects for paying clients.

Think of a bank in 2000. The bank might need an online portal, but it didn't have a large development team. Instead, it hired a consultancy. The consultancy's product was not the software itself - it was the **service** of building software to order. Their success metric was: *did we deliver what the client asked for?*

From this perspective, the values of the Agile Manifesto made sense. For example:

- **Individuals and interactions over processes and tools** fit a consulting context, where client happiness was tied to good communication and a good vendor-client

relationship.
- **Working software over comprehensive documentation** reassured clients that they would actually get usable code, the output.
- **Customer collaboration over contract negotiation** encouraged a flexible procurement with paying clients.
- **Responding to change over following a plan** allowed engagements to be grounded in Transparency, Inspection, and Adaptation. Put more simply, it requires staying aware of emerging changes and adjusting what is intended to be built based on what has been learned.

Notice the orientation: everything is about delivering what the client wanted. In other words, the approach was **output-focused**. The consultancy's job ended with delivering the requested output; whether that output actually improved the bank's business outcomes was left entirely to the bank.

Today, sadly, many companies outsource even more parts of their digital product development; essentially entrusting their corporate brains and backbone to someone else. For some, this could work, even in a long-term vendor relationship. However, for many successful companies, the way of thinking has changed. Critical products are no longer seen as projects to be delivered and then **kept alive or, worse, forgotten**. Instead, organizations expect continuous evolution, learning, and adaptation of their products, often referred to as a **product-operating-model** [16]. Whether built in-house or with external partners, teams are now expected to go beyond output and take responsibility for **outcomes** and, ultimately, **business impact**.

This shift also requires explicit Product Strategy: a clear direction of travel that explains who the product is for, what problem it solves, and how it creates advantage. Product Strategy is essential, but it will be covered in a separate SGEP document because it deserves its own depth.

The Shift From Output to Outcome

Shifting from an output-focused project delivery model—where success is measured by on-time, on-scope, on-budget delivery—to an outcome-oriented product operating model enables organizations to focus less on completed activities and more on the changes in behavior or conditions that indicate meaningful value realization and learning.

Leading indicators measure actions or conditions that are used to forecast future success—they provide signals about progress but do not predict outcomes.
Lagging indicators measure results that have already happened — they show the actual impact of past actions.

Inputs are often shown as the first step. However, something happens before that first step is taken, the motivation to go through the effort. There could be a theory or a result to achieve [31]. Essentially, there are general assumptions about a better future.

- **Assumption** — In the product world, it is about the challenges people have, whether they complain about them or not. The things that people love and de-

sire. These ‘needs’, combined with an understanding of the technologies used to build products and deliver services, form the seeds of innovation. Solving a problem that people already recognize - using a solution they can readily imagine - is one thing. True innovation is meeting a need people did not realize they had, with a solution they could not have anticipated. This is often described as a core driver of personal and business success.

Following through, let’s take a closer look at the value chain, which can be seen as a feedback loop with various learning exit points and starting points, not necessarily linear.

{{< figure src=“images/value-chain-feedback-loop.png” alt=“Value Chain Feedback Loop” caption=“The value chain as a feedback loop showing assumptions, inputs, activities, outputs, outcomes, and impact” >}}

- **Inputs** — Skills, knowledge (including learning from past decisions like outcome, impact, and experiment results), budget, working hours, and tools such as laptops, AI, software, and systems that enable the work. Also, feature requests, to-do lists, action plans, Increments, standards, and sometimes even bug reports can be considered as Inputs. Context matters. Inputs are easy to control.

Inputs, activities, outputs, outcomes, and impact often take into account some assumptions, such as what we observe in the world. The challenges people have, whether they complain about them or not. The things that people love and desire. These needs, combined with what we understand about the technology we use to build things and create services, are the seeds of innovation. It’s one thing to solve a problem people know they have with what they imagine as a solution. It’s true innovation: fulfilling a need they didn’t know they had with a solution they could never have imagined. This is the real secret to personal and business success.

- **Activities** — Talking, meetings, preparing, coding, testing, reducing technical debt, cooperating, and prioritizing. These are the day-to-day actions teams perform. Activities can be controlled.
- **Outputs** — Results from a team’s activities that can be directly produced and measured. Features, to-do lists, action plans, Increments, standards, and sometimes even bugs. Outputs are tangible and easy to measure, but often mistaken for success. They demonstrate feasibility, which can guide the selection of the next inputs. Outputs can be influenced.

Up to this point, the focus has been on ‘leading indicators’ (not useful for forecasting outcomes, so not necessarily ‘leading’).
The following section addresses ‘lagging indicators’.

- **Outcomes** — Measurable changes in customer (including but not limited to users, choosers, internal customers, and other stakeholders) behavior or desired experience — like saving time, enjoying more reliable services, or achieving a smoother journey. Outcomes reveal whether outputs actually made a difference. Outcomes can be anticipated but are not predictable because they are emergent and often serendipitous. Learning from Outcomes may inform Assumptions or Input.

- **Impact** — The long-term organizational effects: revenue growth, improved ROI, cost savings, reputation, or market share. This is what secures survival and long-term competitiveness. For not-for-profit organizations or government institutions, impact can be measured by observable changes, such as a safer, happier public or a cleaner, more sustainable environment. Impact can only be observed. And it takes a cumulative outcome change over time to really have an impact. The impact informs future assumptions.

Bakery Analogy. The bakery wants to be recognized as having the best cake in town, therefore it bakes and sells cakes: assumptions (baker's skills, high-quality ingredients, sufficient demand), inputs (flour, milk, mixer, work hours, recipes) → activities (preparing ingredients, mixing, baking, glazing) → output (cake) → outcome (people enjoy it, go for seconds) → impact (loyal customers, profit, recognition as the cake place in town). Without impact and customer outcomes, the bakery might be busy baking but have shelves full of unpurchased, unenjoyed cakes, and ultimately not achieve the objective and possibly go bankrupt.

In software, teams often stop at the cake. Features are released and labeled as successful. But unless users benefit and are satisfied, and unless the organization benefits, the work has little value.

This also explains why so many "digital transformations" fall short. Companies adopt agile practices, speed up delivery, and push out more features - yet business metrics remain flat. What's missing is the focus on **clear objectives (assumptions)**, **inputs, outcomes, and impacts**. Jeff Bezos was famously known to emphasize **inputs** because they are easier to control - such as building a great team with the right tools, skills, and work environment [20]. The real shift comes when organizations move beyond a project-operating-model, which measures success by output, to a product-operating model, which measures success by what truly changes for customers and the business.

Be aware that outcomes and impacts are not automatically positive. A new feature might succeed in changing customer behavior, but not in the way the team intended. For example, a social media platform introduces autoplay videos. The intended outcome was that user engagement would rise immediately as people spent more time scrolling. However, the change also increased data usage, drained phone batteries, and frustrated users. Over time, complaints grew, and the app rating fell. User frustration and complaints are still outcomes, but negative ones. This causes failure demand, the opposite of value demand [19].

The same applies at the impact level. An aggressive growth strategy might increase revenue in the short term, but it can also damage brand reputation or create unsustainable technical debt. In these cases, the outcome is real, and the impact is measurable, but both are harmful to the organization's long-term health.

A strong product focus based on a product-operating-model addresses trade-offs, tensions, and non-negotiables between short-term wins and long-term consequences.

Acknowledging that outcomes and impacts can be negative keeps teams honest. It is a reminder that success is not defined by change alone, but by desirable change,

experience, and the organization's resilience.

As Annie Duke says in her work on decision-making, decisions should be judged not only by the outcomes they produce, but also by the quality of the process that led to them [7]. A poor decision can occasionally lead to a lucky win, while a sound decision may still result in a setback. However, organizations that consistently base decisions on a transparent process that combines sound reasoning, evidence, and even counterintuitive psycho-logic [28-30] achieve a more durable impact [27]. This allows for objective decision assessment and the resulting process.

At work, you may ask the following questions:

- What customer behavior should change, what might be directly observed? → **the customer outcomes and signals**
- How can the achievement of the outcome be verified? → **the product measures and signals**
- How can the achievement of the desired impact be verified?? → **the company measures and signals**
- What colleague behavior should change, what might be directly observed? → **the work climate outcomes and signals**
- What system behavior should change, what might be directly observed? → **the system of work outcomes and signals**

From Output to Outcome

The Scrum Guide Expansion Pack (SGEP) [14] sharpens this distinction. It introduces the concepts of **Definition of Output Done** and **Definition of Outcome Done**.

- The **Definition of Output Done** ensures that the Increments the Scrum Team creates meet quality standards — good engineering practices, testing, documentation, security (and other non-functional requirements), compliance & regulations, etc.
- The **Definition of Outcome Done** goes further. It asks whether the released Product achieved the intended customer change and business result. Only when outcomes are validated to a sufficient degree to support a decision can the work be considered truly "done." For example: In what ways was customer anxiety about the product's security reduced? How was the product's market position strengthened? How has the market share increased?

The focus on **outcome done** moves **impact** into the spotlight. Without organizational impact — growth, sustainability, resilience — there is no long-term success. Companies may run many Scrum Teams, deploy dozens of features, and still fail in the market if impacts are absent. A major global phone provider's decline is a classic example: outputs kept coming (great phones), but customer outcomes declined (nobody wanted them anymore), and the business impact turned negative.

Outcome-focused Scrum means:

- Scrum Teams are not just builders; they are **stewards or even explorers of value**.

- Product Owners are not Product Backlog secretaries; they are **strategic leaders**.
- Leaders are not **funders of projects**; they are **investors in outcomes** and, ultimately, in product impact.
- When evidence of value realization likelihood is not convincing, they gather quantitative and qualitative evidence on whether features are worth building (e.g., vibe coding, discovery research, customer discovery), but they are aware that the best feedback is result feedback when outputs are released and enabled.

This is what turns Scrum Teams into true committed, even missionary teams - driven by a direction of travel, accountable for outcomes, and committed to long-term impact.

The Learning Loop – Linking Work to Value

A product's journey can also be understood through the (adapted) PDSA cycle (Plan–Do–Study–Act), a learning loop described by W. Edward Deming [26]:

- **Plan**: Based on stakeholder insights, the team hypothesizes (or has a hunch) that customers want a faster checkout. Research shows a high drop-off rate during payment entry, so the goal is to reduce cart abandonment from 42% to 30% within 2 Sprints.
- **Do**: The team prototypes and delivers a one-click checkout option with saved cards, ensuring compliance and good UX practices.
- **Study**: Telemetry data from A/B-tests shows time-to-purchase has been reduced from 5 minutes to 2 minutes, and Sprint Review feedback confirms positive customer reactions.
- **Act**: The team decides to continue improving the feature, rolling it out more widely while planning the next experiment based on what was learned.

Seen this way, each Product Goal, Sprint Goal, or feature works like its own PDSA cycle — small learning loops that can occur at different levels or run in parallel.

To make this direction visible and discussable, many organizations use a Product Roadmap as an evolving communication tool that links goals, options, and decision points over time. Roadmaps are important, but this document treats them as their own topic, so they will be explained in a separate SGEP document rather than here.

Continuously. For more on (adapted PDSA), see the Planguage and Value Planning document. For more on strategy, see the SGEP strategy document.

Meaningful Checks per Sprint (keeping it real)

To ensure feasibility ("can it be built?") while reducing risk, Scrum Teams integrate small but powerful checks into each Sprint. These prevent wasted effort on ideas that look good in theory but collapse in practice.

- **Spikes & technical prototypes as part of value delivery Product Backlog items**: Short, time-boxed investigations to explore unknowns early.
  Example: Before committing to a new recommendation engine, a team runs a timeboxed spike to compare tools or technologies based on model performance.

- **Skill and capacity review**: Ask whether the team has the right people, skills, tools, and time for the proposed work.
  Example: If a Sprint Backlog includes migrating to a new tool, but the team has no experience with containerization, feasibility declines unless training or external support is secured.
- **Dependency mapping and breaking** (internal and external): Identify and make transparent any blockers across teams, vendors, or systems. Remove if possible.
  Example: The checkout feature depends on a payment service from another company that only updates every few months. If the team doesn't spot this dependency early, it could delay the Sprint Goal.
- **Risk-based extensions**: Extend quality checks to match identified risks.
  Example: For a high-traffic e-commerce site, the acceptance criteria might include "checkout response time < 2 seconds on staging with 1,000 concurrent users." Performance and scalability risks are handled before release, not after. Ideally, addressed with the Definition of Output Done.
- **Go/No-Go experiment criteria before continuing**: Define simple success/failure signals for an idea before investing heavily.
  Example: A landing page prototype is tested with 200 real customers. Criteria: at least 15% sign-up conversion rate, and 60% of customers got motivated for learning. If this 'kill-criteria' is not met, the team pivots before building the full feature. Addressed with the Definition of Outcome Done.
- **Desirability check or product telemetry analysis (user motivation and love):** Validate whether the solution resonates with users, not just whether it solves a problem.
  Example: Before finalizing a new dashboard layout, the team tests whether users find it clearer and more engaging, not just functional. Low emotional engagement signals risk of disuse.
- **Viability check (sustainability):** Ensure the solution makes business sense in practice, e.g., within the business constraints.
  Example: A new AI feature is promising, but legal reviews and infrastructure costs reveal it is not viable within the current budget or compliance environment. Adjustments are made before development.
- **Continuous compliance feasibility**: In regulated industries, add (ideally automated) fulfillment checks around rules and standards.
  Example: A health app feature is tested for GDPR and HIPAA compliance during the prototype stage or at every Sprint, preventing costly redesigns later. Addressed with the Definition of Output Done.
- **Tooling and infrastructure validation**: Ensure environments, test data, and deployment pipelines are in place to support the work.
  Example: Before committing to an AI-powered search, the team verifies that anonymized training data is available and legally permissible for use.

Missionary vs. Mercenary Teams

Marty Cagan and John Doerr popularized the distinction between missionary and mercenary teams [8]. It is a powerful metaphor for understanding the required cultural

shift.

Mercenary teams deliver what they are told to do. They are professional, competent, and focused on execution. But they do not own the outcome. Their job ends with delivering output. They have a delivery mindset.

Missionary teams care deeply about the problem they are solving. They are committed to the product strategy, not just the tasks. They own the full value stream - from assumptions, inputs, and activities through outputs to outcomes and impacts. They constantly ask: Did life improve for the customer? How do we know? Did the needle move for the business? How do we know? They follow a value-oriented mode of operation. And, like missionaries in the real world, they're not always successful. They endure a lot of hardship. But do so because ultimately, having an impact is more important to them than their short-term gains.

Neither metaphor should be taken too literally. Mercenaries are not "bad," and missionaries are not "perfect." But the distinction helps see the kind of ownership and passion required in modern product teams.

From a strategic perspective, mercenary teams are paid to deliver outputs, typically without worrying whether those outputs create meaningful impact. Missionary teams, by contrast, link daily work to strategic outcomes. They embody an emergent, continuous adaptive strategy: constantly adjusting to market signals and ensuring the company's long-term impact remains positive.

A mercenary team might say: "We built the feature as specified."
A missionary team might say: "We improved customer happiness by saving 50% of their time, checkout speed by 30%, which led to 15% more completed purchases."

The difference is profound. One team measures success in output. The other measures success based on outcomes and impact.

This is precisely the stance effective Product Owners take: they frame work as hypotheses tied to value or problems to be solved and steer toward outcomes, not activity [18].

Missionary teams also have a sharper eye on the competition. They are not content to satisfy only internal stakeholders; they monitor the market, compare experiences, and ask: *If competitors released the same feature tomorrow, would the product still stand out?* By linking their **why** to both customer needs and competitive positioning, they ensure their work creates defensible impact rather than fleeting output [9].

Good Scrum Teams act as missionary teams.

Evidence-Based Management (EBM)

How are outcomes and impacts measured in practice? This is where Evidence-Based Management (EBM) comes in. Developed by Scrum.org, EBM [5] provides a framework for managing and improving value delivery.

EBM rests on four Key Value Areas (KVAs):

- **Current Value (CV):** How much value are customers, users, and the organization receiving today?
- **Unrealized Value (UV):** How much potential value is still out there?
- **Ability to Innovate (A2I):** How capable is the organization of improving and adapting?
- **Time to Market (T2M):** How quickly can new value be delivered?

Together, these areas encourage teams to look beyond velocity or throughput (merely activity/output measures, often referred to as vanity metrics). Instead, they link organizational capabilities (A2I and T2M) to market value (CV and UV) through a balanced set of measures that provide a signal via telemetry.

In strategic terms, EBM serves as a feedback loop informing the relationship between strategy deployment and emergent strategy. Traditionally, strategy deployment pushes objectives down the hierarchy: leadership sets impact goals, and teams are tasked with delivering outputs that, ideally, align. Emergent strategy, on the other hand, allows teams to discover outcomes through experimentation, observation, and more, and roll those insights up into the strategy. EBM helps leaders validate whether strategic bets are translating into real impact, while empowering teams to adapt locally [10].

EBM also encourages a shift from circumstantial leading indicators to lagging direct evidence - and highlights the role of assumptions and **inputs** in shaping both.

- A **Goal** is the assumption of an objective to be achieved. It is not yet known whether this is feasible or valuable.
- **Leading indicators** are early signals that inputs and outputs are moving in the right direction—such as higher release frequency after investing in CI/CD or faster experiment cycles after adding UX research. Because they show up quickly, they're useful for adjusting course as conditions change.
- **Lagging indicators** confirm (or not) whether those earlier signals turned into real (delayed) outcomes and subsequent impact - like active-user growth, lower churn, or better ROI. They appear later, giving hard evidence of whether the decisions actually worked**.**

By combining experiments with **input tracking, leading signals, and lagging evidence**, teams can validate whether their efforts are producing outputs, creating outcomes, and generating impacts. This creates a continuous learning environment where inputs are actively adjusted, signals inspected, and impacts measured.

Here too, the **why** for **whom** matters. Evidence without purpose is just data. The strategic role of EBM is to anchor metrics — from inputs through outcomes — to the organization's larger **why**: its vision, its customers, and its competitive context. Without that anchor, teams risk chasing numbers that look good in dashboards but do not strengthen the company's long-term impact.

EBM also relies on clearly stated goals across different horizons—strategic, intermediate, and tactical. These goals provide the context that links evidence to purpose, helping Scrum Teams and leaders decide not just what is happening with the product today, but

whether they are moving the product closer to delivering the outcomes and impacts that matter most.

Usability and Viability Metrics — Beyond Features

To ensure **usability**, Scrum Teams measure:

- User task success rate, time on task, and error counts.
- System Usability Scale (SUS) [21] (a 10-item questionnaire using a 5-point Likert scale) or Usability Metric for User Experience (UMUX) [22]-Lite surveys (a 2-question survey that provides a quick usability signal).
- Accessibility audits as part of quality checks.

To ensure **viability**, leaders and Scrum Teams (esp. Product Owners) track business fitness:

- Customer Lifetime Value (LTV) [23] vs. Customer Acquisition Cost (CAC) [23].
- Gross margin and cost-to-serve.
- Payback period and scalability constraints.
- Regulatory and operational risk exposure, recognising compliance and risk reduction as forms of value.

These metrics link short-term feature delivery to the product's long-term sustainability.

Data Collection and Decision Making

A clear measurement framework helps, but raw **data** is also required. Modern product teams are fortunate to operate in a digital environment where telemetry, the automated collection of usage data, provides a near real-time view of how products are used.

Telemetry [24] answers critical questions, for example:

1. Usage & Adoption

- Which features are used, by whom, and which are never used?
- How frequently are key features used?
- Which user groups behave differently?

2. Engagement & Retention

- How often do users return (daily, weekly, monthly)?
- How long do users stay active before dropping off?
- What actions predict long-term use?

3. Flow & Friction (deeper than abandonment)

- Where do users hesitate, retry, or slow down?
- Which steps cause the most errors or retries?
- Where do users need help or support?

4. Value Realization

- Do users reach the intended outcome?
- How long does it take users to get value for the first time?
- Which behaviors correlate with success?

5. Change Over Time

- How does behavior change after a release?
- Did a new feature replace an old workaround?
- Are users adapting or ignoring the change?

6. Risk & Quality Signals

- Which actions lead to crashes, errors, or support tickets?
- Are there patterns before churn or complaints?
- Which users are at risk of leaving?

7. Business-Relevant Signals

- Which behaviors correlate with upgrades or renewals?
- Which features are used by high-value customers?
- What usage patterns predict willingness to pay?

This matters because not all signals are equally strong. A spike in page views might look promising, but if users abandon the flow before completing a task, the signal is weak. By contrast, sustained daily use, customer referrals, or willingness to pay are strong signals of value.

Retiring unused features is equally important. Every feature carries a maintenance cost. If telemetry shows little adoption, the courageous choice might be to retire it. This creates space for investing in outcomes that matter.

Data collection alone does not create value. Teams must build a discipline of **data-informed** decision-making and integrate it into their product-operating-model. That means decisions are guided by evidence, but not enslaved to it. Numbers lack context; teams bring purpose and vision to interpret what signals really mean. The **why** for the **whom** behind the data turns metrics into actionable insight.

It is encouraged to use both quantitative data (e.g., number of active users, time to complete a task) and qualitative data (e.g., customer feedback, user interviews, or user observations). Together, they provide a fuller picture: the numbers show **what** is happening, while stories and observations explain **why**.

Making Decisions Informed by Evidence

Adaptation without facts is guesswork. Product management requires choices: which customers to prioritize, which bets to fund, which experiments to scale.

Evidence-informed decision-making closes the gap between intuition and reality. It does not mean eliminating judgment; it means grounding judgment in evidence. Be open to psycho-logic: the opposite of a good idea might be another good idea [28];

if every idea is rational, the competition will do the same things. A Product Owner deciding which feature to invest in might ask:

- What data supports this choice?
- What assumptions are being made?
- How will success be known?
- Which experiment could validate the assumption?
- What critical thinking has been applied?

The discipline is simple but powerful: **state assumptions or hunches, collect evidence, inspect results, and adapt**. This creates a feedback loop that replaces opinion-driven debates with learning-driven action. Leaders, especially Supporters [14], have a role too: rather than rewarding certainty, they must create a safe environment for admitting uncertainty and adjusting based on facts.

Without this discipline, organizations fall into two popular traps (there could be more). One is **HiPPO decisions** [15]— following the Highest Paid Person's Opinion. The other is **analysis paralysis** — drowning in dashboards without making choices. Fact-or-figure-informed decision-making strikes a balance: enough evidence to reduce risk and enough courage to act. It also helps Scrum Teams minimize cognitive and organizational biases that can distort judgment.

One of the most useful ways to set goals is to identify value that likely exists but hasn't been captured yet. This means making educated guesses about where the biggest gaps are between what customers get today and what they still need. By comparing current customer outcomes with unmet needs (and what competitors offer), teams can spot the most promising opportunities. For example, if customers use the product frequently but repeatedly complain about a missing capability, that's a strong signal that the capability is worth pursuing. Used this way, these educated guesses help focus investment where the potential return is highest. Human decision-making is often based on emotions and biases - not necessarily always logical. While making a decision, these human traits should be recognized, even leveraged.

Product Risk Management

Another reason that project-operating-models fail is a classical approach to risk management. Risk is often treated as something to eliminate with upfront analysis and control. But in complex domains, most risks cannot be fully known in advance. Risk is good when it is made visible early and used to guide learning. Agile approaches manage risk differently: by delivering in small Increments, proactively prioritizing risk reduction, testing assumptions early, and using feedback to correct course. Leveraging those as enabling constraints allows Scrum Teams to reduce uncertainty while still moving forward. Managing risk this way keeps organizations adaptive, prevents costly failures, and improves the chances that money, effort, and other resources are spent on what truly matters. This is an emergent strategy applied (also known as a continuous adaptive strategy).

An Agile approach to risk management has always involved:

- **Small Increments:** Reducing exposure by releasing frequently.
- **Continuous discovery with early validation:** Build to learn with prototypes or experiments before scaling.
- **Feedback loops:** Using telemetry and customer feedback as continuous risk signals.
- **Portfolio thinking:** Diversifying bets (with "kill criteria") rather than putting all money and efforts into one initiative.

The **why** here is 1). survival in uncertainty and 2). excelling in value improvement. By managing risk adaptively, organizations avoid the false comfort of static plans. They learn faster than competitors and adjust before risks become existential. In practice, this means risk registers and upfront mitigation plans are replaced with living experiments and **signal-based evidence-informed** decisions.

Adaptive risk management turns uncertainty from a threat into an asset. Variation in outcomes is not feared - it is the source of learning that guides future strategy; it is a competitive advantage.

Product Engineering Practices – Building for Feasibility and Risk Reduction

In digital products, strong engineering practices reduce product risk and ensure feasibility:

- **Test automation:** Using software tools to automatically run tests, check results, and report issues - making testing faster, repeatable, and more reliable
- **CI/CD pipelines:** Frequent integration and deployment reduce release risk.
- **Trunk-based development:** Keeps work integrated, reducing merge conflicts.
- **Feature flags & canary releases:** Enable safe experimentation and gradual rollouts.
- **Observability (SLO/SLA/SI) [25]:** Telemetry, error rates, and performance signals ensure quality **in use**.
- **Security by design:** Embedding security testing and compliance checks into the Definition of Output Done.

These practices tie engineering excellence directly to value delivery, turning quality into a risk-reduction tool. Similar principles apply to non-IT products as well: in domains such as healthcare, medical devices, and pharmaceuticals, practices such as automated quality checks, small-batch experimentation, and regulatory compliance embedded early in the process serve the same purpose: reducing risk.

Compliance and Regulations

Compliance is often seen as an obstacle to agile product management. In regulated industries -finance, healthcare, energy - teams may assume they cannot be agile because every release must pass strict audits. But compliance and agility are not opposites. Done well, they reinforce each other. For example, change agents could foster regular, intentional collaboration between the product team and single points of contact to determine how each set of compliance risks would enable continuous compliance at pace.

The **Definition of Output Done** in the SGEP ensures that Increments meet the agreed quality standards. In regulated environments, these standards must include **compliance and regulatory checks**. A feature that works for customers but violates data privacy rules is not truly "output done."

How does this work in practice?

- **Embed compliance** into the Definition of Output Done. This shifts compliance from a separate gate at the end to an integral element of the quality of every Increment in every Sprint. It becomes ingrained in the Product Developer's daily behavior.
- **Automate evidence collection.** Audit trails, test reports, and documentation should be generated as a byproduct of delivery, not as a scramble before release.
- **Use inspection and adaptation.** Regular reviews with compliance officers ensure evolving rules or deviations are addressed continuously, not retroactively.
- **Share the Definition of Output Done** at every Sprint Review. Build trust by being transparent about what "output done" means in terms of quality. *Do not demonstrate a not "output done" feature — that betrays trust.*

Compliance is about **earned trust**. Customers trust that their data is safe, regulators trust that companies act responsibly, and leaders trust that teams deliver without exposing the organization to existential risks. Agility and compliance are not mutually exclusive; done right, they can help build a **competitive advantage**.

Ethics Guardrails – Working Within Boundaries

Scrum requires teams to create products **within the ethical boundaries of their context**. This means teams must actively prevent harm while seeking outcomes and impacts. Guardrails include:

- **No dark patterns:** Avoid manipulative designs that trick users into actions.
- **Data minimization:** Collect only what is necessary, with explicit user consent.
- **Fairness and bias checks:** Ensure features and algorithms treat groups equitably.
- **Transparency:** Make terms, data use, and risks visible to users and stakeholders.
- **Accessibility as default:** Products must be usable by people of diverse abilities.
- **Safety reviews:** Include ethical and safety checks.
- **Responsible AI use:** Validate AI outputs for fairness, accuracy, and explainability; ensure humans remain accountable for critical decisions.

Embedding these into the **Definition of Output Done** and **Definition of Outcome Done** ensures Scrum Teams deliver value **responsibly** and **sustainably**.

How to Put It into Action – Practical Guidance for Organizations

For Teams – How to shift from output to outcome

- **Claim autonomy with intent:** Define clear Sprint Goals and agree as a Scrum Team how you'll reach them. Own the outcome and measure it, don't wait for step-by-step instructions.

- **Start with outcome & own the whole chain:** Map how your work links from task → feature → outcome → impact. Build telemetry into your product. Regularly check if you're still moving the needle.
- **Practice discovery and measurement:** Run small experiments (A/B tests, prototypes, user interviews). Track leading indicators (outputs), as well as lagging indicators (outcome/impact), and use results to adapt [11].
- **Be explicit about usability:** Track task success rate/time on task, run SUS/UMUX-Lite surveys, and verify accessibility in every Sprint Review.
- **Feasibility first:** Use spikes to derisk unknowns; review skills/dependencies each Sprint.

For Leaders – How to enable outcome thinking

- **Fund outcomes, not projects:** Allocate budgets to outcome-related Product Goals, not fixed feature lists. Consider rolling budgets [12, 32].
- **Invest in stable Scrum Teams:** A group of people working together is not a team; a team is an emergent property. Build great Scrum Teams and keep them together so they can build your products. Stable, empowered teams have far higher chances of outcome predictability. If there is unavoidable or deliberate dynamism in team membership, consider intentionally adopting Dynamic Reteaming [33].
- **Use metrics intentionally:** Track metrics that reflect outcome and impact as well as your internal capabilities.
- **Balance deployment with emergence:** State the desired impact clearly, but let Scrum Teams experiment and discover how best to achieve it. Review and adjust strategy when new evidence emerges.
- **Watch viability:** Review LTV (Lifetime Value)/CAC (Customer Acquisition Cost), cost-to-serve, and payback alongside other metrics each quarter.

For Customers – How to experience the difference

- **Co-create solutions:** Participate in Refinements, and provide feedback through regular Sprint Reviews, user testing, and surveys. Expect to see changes in response.
- **Look for evidence of learning:** notice whether your pain points are being resolved and whether improvements are sustained.
- **Build trust over the long term:** Stick with products and teams that demonstrate consistency in learning, adapting, and delivering meaningful outcomes.

Pitfalls to Avoid – And How to Steer Clear

- **Vanity metrics:** Instead of tracking the number of downloads or story points, track retention, engagement, or customer value delivered.
- **Ritualistic Agile (often called Cargo-Cult):** Tie every Sprint Goal back to a Product- or Strategic Goal. Ask, *"How does this Sprint move us closer to impact?"*

- **Short-termism:** Balance quick wins with investment in sustainable technology, user trust, and long-term goals.
- **Losing sight of the why:** Start each Product Backlog Item refinement by asking, *"Which outcome does this serve?"* and *"How will success be measured?"*
- **Risk blindness:** Break work into small bets, run time-boxed experiments, and surface risks early. Use learning reviews to adapt rather than assuming the plan holds.
- **Ethical pitfalls:** Avoid dark patterns and addictive loops; measure **healthy** engagement (value-in-use) rather than just time-on-site.

Conclusion

The Agile Manifesto was of its time - success was defined as delivering software output to clients. But the world has changed; the world of product development learned a lot in the last few decades. However, there is no single best practice; perspective matters. At a minimum, they must connect assumptions to inputs, inputs to activities, activities to outputs, outputs to outcomes, and outcomes to impacts.

Impact is what companies ultimately need for survival. **Outcomes** are about customer change, but impact is about whether the business thrives. It is the link between agility and strategy, between experiments and long-term direction. Companies that ignore impact risk become busy but irrelevant. Companies that embrace outcome-driven impact become resilient, adaptive, and sustainable.

This requires a cultural shift — from **mercenaries to missionaries**, from **project thinking to product thinking**, from **circumstantial indicators to evidence-informed management**. It also requires leaders to balance **strategy deployment** with **emergent strategy**; establish organizational transfer of learning and multi-learning [13] by setting ambitious goals, while allowing self-managing Scrum Teams to discover the best path to achieve them.

The path forward is clear:

- Establish a **product approach**.
- Consider **EBM** or other approaches (e.g., adapted PDSA) to validate assumptions, measure outcomes and impacts, and make informed decisions.
- Apply **SGEP** to distinguish between "output done" and "outcome done"
- Consider adopting a continuous adaptive strategy.
- Build Scrum Teams that **own the entire product value stream**.
- Treat **impact** as the ultimate measure of success.

In short: **Stop celebrating cake recipes.** Start measuring how many people enjoyed the cake, came back for more, and kept the bakery thriving.

The most resilient companies keep the **why** and for **whom** of their strategy at the center. They manage risk not by pretending uncertainty can be eliminated, but by learning quickly and cheaply. They see variation as a benefit, and they understand competition not as a threat but as a forcing function to stay focused on outcomes and impacts that

truly differentiate. When combined, these practices turn Scrum from a delivery engine into a **strategic advantage**.

Only then can Scrum be said to have fulfilled its promise - not just to build software, but to maximize customer value and create **lasting organizational impact**.

References

[1] Beck, K. et al., 2001. Manifesto for Agile Software Development. Available at: https://agilemanifesto.org (Accessed: 20 September 2025).

[2] Beck, K., 2010. To Agility, and Beyond. Startup Lessons Learned Conference. Available at: https://www.youtube.com/watch?v=d4qldY0g/_dI (Accessed: 20 September 2025).

[3] Gothelf, J. and Seiden, J., 2021. Lean UX: Designing Great Products with Agile Teams. 3rd ed. Sebastopol, CA: O'Reilly Media.

[4] Ries, E., 2011. The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. New York: Crown.

[5] Scrum.org, 2024. Evidence-Based Management™ (EBM). Available at: https://www.scrum.org/resources/evidence-based-management (Accessed: 20 September 2025).

[6] Denning, S., 2018. The Age of Agile: How Smart Companies Are Transforming the Way Work Gets Done. New York: AMACOM.

[7] Duke, A., 2018. Thinking in Bets: Making Smarter Decisions When You Don't Have All the Facts. New York: Portfolio.

[8] Cagan, M., 2020. Missionaries vs. Mercenaries. Silicon Valley Product Group. Available at: https://www.svpg.com/missionaries-vs-mercenaries/ (Accessed: 20 September 2025).

[9] Cagan, M., 2018. Inspired: How To Create Products Customers Love. 2nd ed. Hoboken, NJ: Wiley.

[10] Schwaber, K. and Sutherland, J., 2020. The Scrum Guide. Available at: https://scrumguides.org (Accessed: 20 September 2025).

[11] Knapp, J., Zeratsky, J. and Kowitz, B., 2016. Sprint: How to Solve Big Problems and Test New Ideas in Just Five Days. New York: Simon & Schuster.

[12] Verma, R. et al., 2021. 10 Steps to Integrate Evidence-Based Management with Scrum in 21 Days or Less. Scrum.org. Available at: https://www.scrum.org/resources/10-steps-integrate-evidence-based-management-scrum-21-days-or-less (Accessed: 20 September 2025).

[13] Takeuchi, H. and Nonaka, I., 2014. The New New Product Development Game. Harvard Business Review. Available at: https://hbr.org/1986/01/the-new-new-product-development-game (Accessed: 20 September 2025).

[14] Sutherland, J., Jocham, R. and Coleman, J., 2025. Scrum Guide Expansion Pack. Available at: https://scrumexpansion.org (Accessed: 20 September 2025).

[15] Kaushik, A., 2009. Web Analytics 2.0: The Art of Online Accountability and Science of Customer Centricity. Indianapolis: Sybex.

[16] Cagan, M., 2023. The Product Operating Model: An Introduction. Silicon Valley Product Group. Available at: https://www.svpg.com/the-product-operating-model-an-introduction/ (Accessed: 20 September 2025).

[17] Seiden, J., 2019. Outcomes Over Output: Why Customer Behavior is the Key Metric for Business Success. New York: Sense & Respond Press.

[18] McGreal, D. and Jocham, R., 2018. The Professional Product Owner: Leveraging Scrum as a Competitive Advantage. Boston, MA: Addison-Wesley Pearson.

[19] Cusumano, M.A. and Selby, R.W., 1995. Microsoft Secrets: How the World's Most Powerful Software Company Creates Technology, Shapes Markets, and Manages People. New York: Free Press.

[20] Amazon.com, Inc., 2010. 2009 Letter to Shareholders. Available at: https://www.sec.gov/Archives/edgar/data/1018724/000119312510082914/dex991.htm (Accessed: 20 September 2025).

[21] Brooke, J., 1996. SUS: A 'quick and dirty' usability scale. In: P.W. Jordan, B. Thomas, B.A. Weerdmeester and A.L. McClelland, eds. Usability Evaluation in Industry. London: Taylor & Francis, pp.189-194.

[22] Finstad, K., 2010. The Usability Metric for User Experience. Interacting with Computers, 22(5), pp.323-327. Available at: https://doi.org/10.1016/j.intcom.2010.04.004 (Accessed: 20 September 2025).

[23] Gupta, S. and Lehmann, D.R., 2005. Managing Customers as Investments: Strategic Value of Customers in the Long Run. Upper Saddle River, NJ: Wharton School Publ.

[24] Riedesel, J., 2021. Software Telemetry: Reliable Logging and Monitoring. Manning Publications, New York.

[25] Beyer, B., Jones, C., Petoff, J. and Murphy, N.R., 2016. Site Reliability Engineering: How Google Runs Production Systems. Sebastopol, CA: O'Reilly Media.

[26] Deming, W.E. (1986) Out of the crisis. Cambridge, MA: Massachusetts Institute of Technology, Center for Advanced Engineering Study.

[27] Pfeffer, J. and Sutton, R.I. (2006) 'Evidence-based management', Harvard Business Review, 84(1), pp. 62-74.

[28] Sutherland, R. (2021) Alchemy. London: WH Allen. Available at: https://www.penguin.co.uk/books/430379/alchemy-by-rory-sutherland/9780753556528 (Accessed 27 December 2025). To keep the insights from 'Alchemy' active and integrated into your strategic thinking, consider revisiting one rule of Alchemy each quarter. This

cyclical practice can ensure that these innovative ideas remain a living part of your strategy, continually inspiring fresh approaches and guiding decision-making.

[29] MadFest London (2025) Rory Sutherland on why being delightfully less wrong beats being boringly right. Available at: https://www.madfestlondon.com/insights/articles/rory-sutherland-on-why-being-delightfully-less-wrong-beats-being-boringly-right/ (Accessed 27 December 2025).

[30] 42courses (2023) Rory Sutherland's '11 Rules of Alchemy'. Available at: https://www.42courses.com/blog/home/rory-sutherlands-11-rules-of-alchemy (Accessed 27 December 2025).

[31] W.K. Kellogg Foundation (2004) Logic Model Development Guide. Battle Creek, MI: W.K. Kellogg Foundation. Available at: https://wkkf.issuelab.org/resource/logic-model-development-guide.html (Accessed: 14 January 2026).

[32] Bogsnes, B. (2023) Rethinking how we manage organizations in a post-industrial world, Beyond Budgeting. At: https://bbrt.org/wp-content/uploads/bb_principles.pdf (April 5, 2023).

[33] Helfand, H. (2019) Dynamic Reteaming: The Art and Wisdom of Changing Teams. 2nd edn. Walnut Creek, CA: O'Reilly Media.