PROJECT AND PROGRAM MANAGEMENT

Lean, Agile and Iterative

Beck · Sutherland & Schwaber · Anderson · Reinertsen

PAPER 02 / 03

A Furtherance reference compendium on the intellectual traditions behind organisational design, leadership, execution, and culture.

IN THIS PAPER

Where goals and methods are uncertain, plan-driven delivery struggles. This paper covers the agile and lean reaction — delivering through short iterations, flow and feedback rather than a fixed up-front plan. The question it answers: how do you deliver complex, uncertain work when you cannot specify it all in advance?

Beck brought disciplined engineering and continuous delivery; Sutherland and Schwaber gave agile its dominant empirical framework in Scrum; Anderson applied lean flow to knowledge work through Kanban; Reinertsen supplied the economic and queueing theory beneath it all.

Kent Beck **Extreme Programming, 1999 · Agile Manifesto, 2001**

Software engineering · disciplined agility

CORE THESIS

Embrace change rather than resist it, and make it safe through disciplined engineering and tight feedback. Extreme Programming pushes proven practices to the extreme so that software can be changed continuously and safely.

FRAMEWORK

Practice	What it delivers
Test-driven development	Confidence to change code safely
Continuous integration	Frequent, small, verified merges
Pair programming	Built-in review and shared knowledge
Small releases + refactoring	Fast feedback and a clean, evolvable design

METHODOLOGY

Engineering practice, codified and later generalised through the Agile Manifesto's values and principles.

SIGNIFICANCE AND CRITIQUE

The engineering backbone of agile — test-driven development and continuous integration are now mainstream. The critique: the practices are demanding, and their software origin does not always translate cleanly to other work.

Sutherland & Schwaber **Scrum, 1995–**

Agile · the empirical process framework

CORE THESIS

Complex product work should be run **empirically**: deliver in short, timeboxed **sprints**, then **inspect and adapt**. Transparency, inspection and adaptation replace the illusion of a perfect plan.

FRAMEWORK

Scrum element	Purpose
Roles	Product Owner, Scrum Master, Developers
Events	Sprint, planning, daily scrum, review, retrospective
Artefacts	Product backlog, sprint backlog, increment
Empiricism	Transparency, inspection, adaptation each sprint

METHODOLOGY

A lightweight framework derived from practice in complex product development.

SIGNIFICANCE AND CRITIQUE

The dominant agile framework worldwide. The critique: it is often implemented mechanically – ceremony without empiricism – and scaling it across large organisations is genuinely hard.

David Anderson **Kanban, 2010**

Lean flow · evolutionary change

CORE THESIS

Apply lean **flow** to knowledge work: **visualise the workflow**, **limit work-in-progress**, and **manage flow**. Kanban is evolutionary – start where you are, and improve continuously – rather than imposing a new process wholesale.

FRAMEWORK

Practice	Effect
Visualise the work	Make the flow and bottlenecks visible
Limit work-in-progress	Finish before starting; expose constraints
Manage flow	Optimise for smooth, fast throughput
Flow metrics	Lead time and throughput (Little's Law) guide improvement

METHODOLOGY

Lean and the Theory of Constraints applied to knowledge work as an evolutionary change method.

SIGNIFICANCE AND CRITIQUE

Less disruptive than Scrum and strong on flow, it suits teams that cannot reorganise wholesale. The critique: being incremental, it can leave deeper structural problems untouched.

Donald Reinertsen **The Principles of Product Development Flow, 2009**

Lean product development · the economics of flow

CORE THESIS

Beneath agile and lean lies **economics** and **queueing theory**. Manage product development as a flow of work through queues: quantify the **cost of delay**, reduce **batch size**, limit work-in-progress, and manage cadence – and make decisions on economic, not proxy, grounds.

FRAMEWORK

Principle	Why it matters
Cost of delay	The single most useful number for prioritisation
Reduce batch size	Smaller batches cut cycle time and risk
Manage queues / WIP	Queues, not capacity, drive most delay (Little's Law)
Cadence and decentralisation	Rhythm and local control speed decisions

METHODOLOGY

Economics and queueing theory applied rigorously to product development.

SIGNIFICANCE AND CRITIQUE

The intellectual foundation beneath lean and agile flow, and the source of cost-of-delay prioritisation. The critique: the material is technical and dense for a general audience.

Synthesis

Contribution	Focus	Key idea	Register
Beck	Engineering	XP; continuous delivery	Practice
Sutherland & Schwaber	Process framework	Scrum; empiricism	Framework
Anderson	Flow	Kanban; limit WIP	Method
Reinertsen	Economics	Cost of delay; queues	Theory

FOR THE OPERATOR

The agile and lean turn replaces the up-front plan with empirical iteration and flow – disciplined engineering and continuous delivery (Beck), an empirical sprint framework (Scrum), flow through work-in-progress limits (Anderson), grounded in the economics of queues and cost of delay (Reinertsen). For the operator: for uncertain, complex work, manage flow and feedback rather than plans, and prioritise by cost of delay. The final paper scales up – to programs, portfolios and megaprojects.