AGILE REVISITED

The growth and change of an incremental and iterative methodology.

INTRODUCTION

Once upon a time, there was a ski resort in Utah. It was 2001, and 17 practitioners/advocates of flexible programming found that they had more in common than not. The result was the **Agile Manifesto**, and the Agile Movement was born. The Agile approach delivers solutions of demonstrated business value, through a collaboration of self-organizing and cross-functional teams. It advocates adaptivity, evolution, and continuous improvement, creating an environment that responds rapidly to change. Not surprisingly, Agile itself has adapted, evolved, and improved.

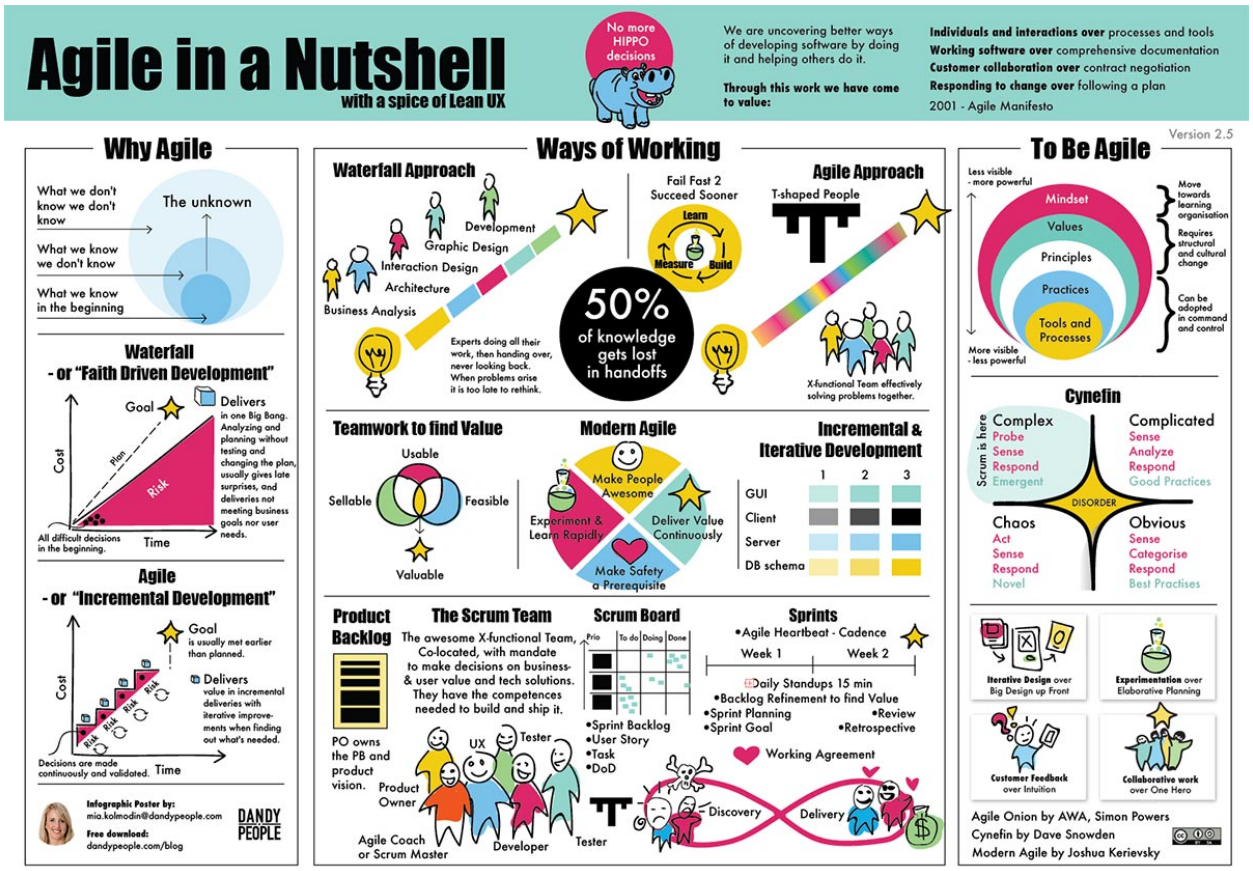
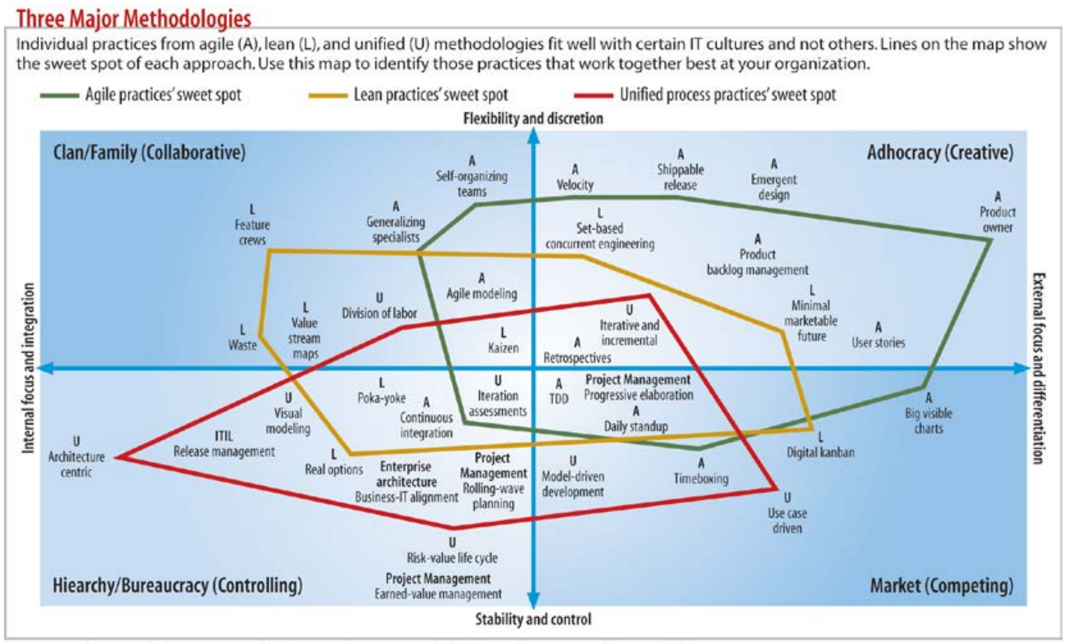

Agile in a Nutshell
with a spice of Lean UX
No more HIPPO decisions
We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:
Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan
2001 - Agile Manifesto
Version 2.5
Why Agile
What we don't know we don't know
The unknown
What we know we don't know
What we know in the beginning
Waterfall - or "Faith Driven Development"
Goal
Plan
Risk
Cost
Time
Delivers in one Big Bang. Analyzing and planning without testing and changing the plan, usually gives late surprises, and deliveries not meeting business goals nor user needs.
All difficult decisions in the beginning.
Agile - or "Incremental Development"
Goal is usually met earlier than planned.
Delivers value in incremental deliveries with iterative improvements when finding out what's needed.
Decisions are made continuously and validated.
Infographic Poster by: mia.kolmodin@dandypeople.com
Free download: dandypeople.com/blog
DANDY PEOPLE
Ways of Working
Waterfall Approach
Business Analysis
Architecture
Interaction Design
Graphic Design
Development
Experts doing all their work, then handing over, never looking back. When problems arise it is too late to rethink.
Fail Fast 2 Succeed Sooner
Learn
Measure
Build
50% of knowledge gets lost in handoffs
Agile Approach
T-shaped People
X-functional Team effectively solving problems together.
Teamwork to find Value
Usable
Sellable
Feasible
Valuable
Modern Agile
Make People Awesome
Experiment & Learn Rapidly
Deliver Value Continuously
Make Safety a Prerequisite
Incremental & Iterative Development
GUI
Client
Server
DB schema
Product Backlog
PO owns the PB and product vision.
The Scrum Team
The awesome X-functional Team, Co-located, with mandate to make decisions on business- & user value and tech solutions. They have the competences needed to build and ship it.
Product Owner
UX
Tester
Agile Coach or Scrum Master
Developer
Scrum Board
Prio
To do
Doing
Done
•Sprint Backlog
•User Story
•Task
•DoD
Sprints
•Agile Heartbeat - Cadence
Week 1
Week 2
Daily Standups 15 min
•Backlog Refinement to find Value
•Sprint Planning
•Sprint Goal
•Review
•Retrospective
Working Agreement
Discovery
Delivery
To Be Agile
Less visible - more powerful
Mindset
Values
Principles
Practices
Tools and Processes
More visible - less powerful
Move towards learning organisation
Requires structural and cultural change
Can be adopted in command and control
Cynefin
Scrum is here
Complex
Probe
Sense
Respond
Emergent
Complicated
Sense
Analyze
Respond
Good Practices
DISORDER
Chaos
Act
Sense
Respond
Novel
Obvious
Sense
Categorise
Respond
Best Practises
Iterative Design over Big Design up Front
Experimentation over Elaborative Planning
Customer Feedback over Intuition
Collaborative work over One Hero
Agile Onion by AWA, Simon Powers
Cynefin by Dave Snowden
Modern Agile by Joshua Kerievsky

Image courtesy of dandypeople.com

Cut to 20 years later, Agile has become the "new traditional" process for development. The main Agile variants (Agile, Scrum, eXtreme Programming) have grown to the point where 91% of US companies are at least partly Agile.

These variants have their own rituals, but still have more in common than not. I like this old study that shows four basic practices at the core of the variants: daily standups, retrospectives, iterative/incremental, and test driven development.


Three Major Methodologies
Individual practices from agile (A), lean (L), and unified (U) methodologies fit well with certain IT cultures and not others. Lines on the map show the sweet spot of each approach. Use this map to identify those practices that work together best at your organization.
Agile practices' sweet spot
Lean practices' sweet spot
Unified process practices' sweet spot
Flexibility and discretion
Stability and control
Internal focus and integration
External focus and differentiation
Clan/Family (Collaborative)
Adhocracy (Creative)
Hiearchy/Bureaucracy (Controlling)
Market (Competing)
A Self-organizing teams
A Velocity
A Shippable release
A Emergent design
A Product owner
L Feature crews
A Generalizing specialists
L Set-based concurrent engineering
A Product backlog management
A Agile modeling
U Division of labor
L Value stream maps
L Waste
L Kaizen
U Iterative and incremental
A Retrospectives
L Minimal marketable future
A User stories
U Visual modeling
L Poka-yoke
A Continuous integration
U Iteration assessments
A TDD
Project Management Progressive elaboration
A Daily standup
A Big visible charts
U Architecture centric
ITIL Release management
L Real options
Enterprise architecture Business-IT alignment
Project Management Rolling-wave planning
U Model-driven development
A Timeboxing
L Digital kanban
U Use case driven
U Risk-value life cycle
Project Management Earned-value management

Data: Application of *Diagnosing and Changing Organizational Culture*, Cameron and Quinn, 2006

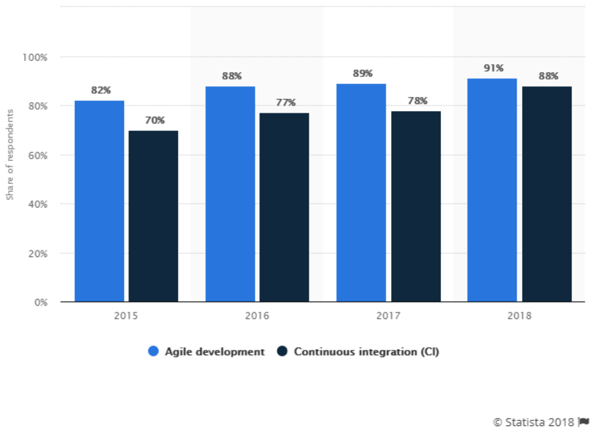
THE CHILDREN OF AGILE

There have been a number of methods that have influenced, and been influenced by, Agile: DevOps, Theory of Constraints, Lean, Phoenix Project, Cynefin, SAFe, to name a few. I will explore what they are, and when they should be used below.

DevOps / Continuous Integration – Agile introduced the theory of iterations (bi-weekly, monthly, etc.), but found that delivering into (non-Agile) production was a challenge. DevOps started as a method for explicitly bridging the rapid release cycle of Agile safely into production, and is now a sub-discipline of its own. Continuous Integration is the discipline and toolset supporting daily builds of developed code, feeding into DevOps.

Good for: Rapid, lightweight applications like Web, mobile

Bad for: Waterfall enterprises

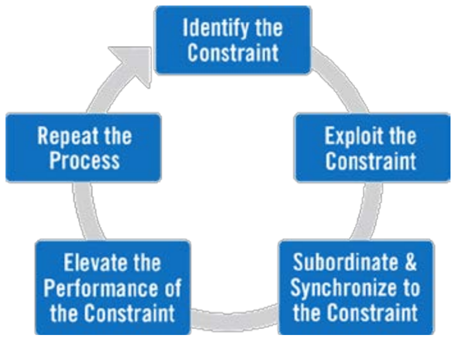


Lean – Adapted from the Toyota Production System from the manufacturing world, Lean was a method based on waste minimization without sacrificing productivity. Lean also focused on unevenness in workloads which tends to be a cause of bottlenecks in production settings. Lean was then adapted to software development and considered a subculture within the Agile community. It is based on seven principles:

1. Eliminate waste
2. Amplify learning
3. Decide as late as possible
4. Deliver as fast as possible
5. Empower the team
6. Build integrity in
7. See the whole

Good for: Iterative, incremental improvement of existing processes

Bad for: Reimagining a new process or workflow


Identify the Constraint
Exploit the Constraint
Subordinate & Synchronize to the Constraint
Elevate the Performance of the Constraint
Repeat the Process

Theory of Constraints – This theory addresses bottlenecks in production settings and which bottlenecks cause the most delays. The challenges are the usual: time and resources. The idea behind the theory of constraints is that every process has a constraint/bottleneck. Focusing effort on clearing/improving this constraint will be the fastest and most effective path to improved results. *(Image courtesy of leanproduction.com)*

Good for: Production, Operating environments

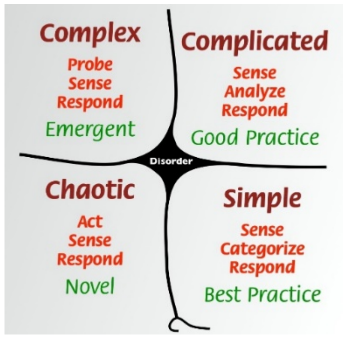
Bad for: Panicked environments not interested in learning new things

Phoenix Project - Okay, so this is a book, not a method. But it explores a mashup of the Agile, Lean, and Theory of Constraints. It proposes combining elements from the different methodologies to create a custom cocktail for your projects. The Phoenix Project is a novel that introduces and explores "The Three Ways" to do just that:

1. Emphasize the performance of an entire system as opposed to the performance of a single department,
2. Create feedback loops for easier communication and correction, and
3. Finally create a culture of continual experimentation.

Good for: Everything

Bad for: Something not covered in the story line of the book

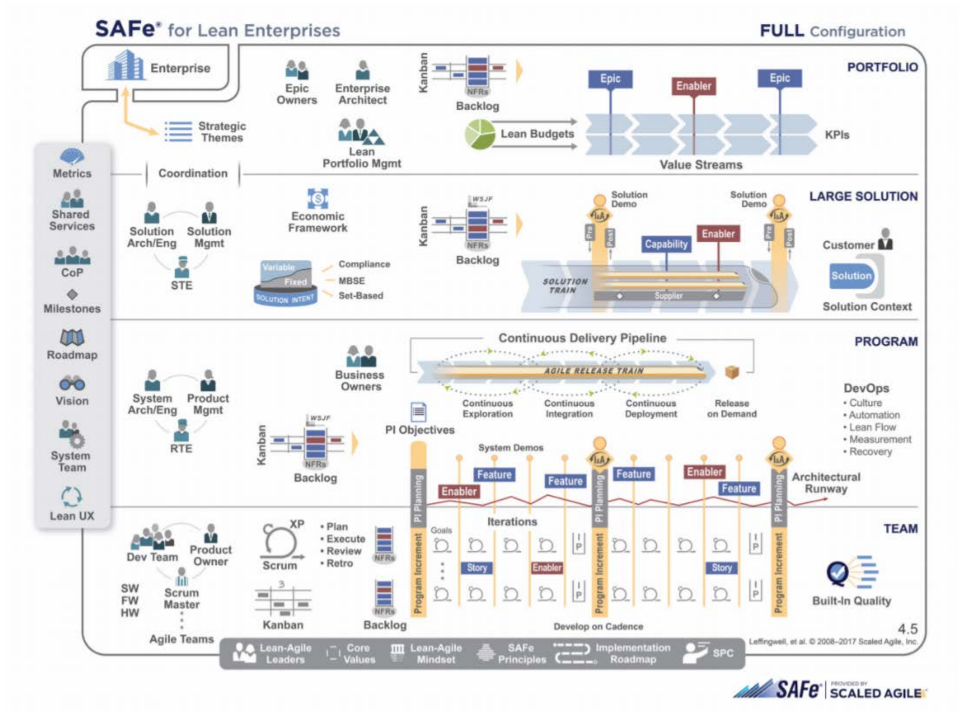


Cynefin – Cynefin (pronounced KUN-iv-in, a Welsh word meaning "habitat") is an interesting framework initially created by David Snowden at IBM Global Services in 1999. It focuses more on how you make decisions for a company and how the ideal state is determined by balancing both company requirements and people's behavior. The framework offers five decision-making "domains" that help managers perceive environments and understand their corporate behavior. *(Image courtesy of Wikipedia)*

Good for: Crisis Management

Bad for: Dogmatic structured environments

SAFe – Agile began and grew in the world of Web and smartphone applications, only later addressing legacy applications and large-scale programs. SAFe (Scaled Agile Framework) is meant to guide companies in scaling lean and agile projects to accommodate longer planning horizons and multiple teams. SAFe generally takes a "Big Picture" look across multiple workflows or workstreams. SAFe took the basic principles of Agile and added in product management, governance, and multiple program and development teams working simultaneously. This has slowly become recognized as the more common approach to scaling development projects. SAFe is a broad framework; expert practitioners typically "cut it down" for what is needed for particular programs.



SUMMARY

A quick Google search will render hundreds of thousands of results on everything from dissecting the Agile Principles and Frameworks to a myriad of success and failure stories. This TIP outlined a few of these frameworks. Here is the guidance compiled into a single table for comparison.

Child	Means	Good For	Bad For
DevOps/Continuous Integration	Bridging Agile development with non-Agile Operations	Rapid, lightweight applications like Web, mobile	Waterfall enterprises
Lean	Based on waste minimization without sacrificing productivity	Iterative, incremental improvement of existing processes	Reimagining a new process or workflow
Theory of Constraints	Iteratively remove the most business-impactful bottlenecks	Production, Operating environments	Panicked environments not interested in learning new things
Phoenix Project	Pragmatic “method by example” blending elements of Agile, Lean, DevOps, ToC	Everything ☺	Something not covered in the story line of the book
Cynefin	Match your response / approach to the environment and the goal	Crisis Management	Dogmatic structured environments
SAFe	Accommodates longer planning horizons and multiple teams, “Big Picture” look	Enterprise-wide programs	Small, quick projects (e.g. mobile apps)

Remember, each project requires a different framework and each framework requires a different implementation. Your single most important step for successfully navigating and executing that implementation is formal training and working with a Coach who has implemented Agile many times in the real world.

Trexin is here to help.

This TIP was written by Ridhi Patel, a Principal at Trexin. Ridhi welcomes comments and discussion on this topic and can be reached at ridhi.patel@trexin.com.