1.3.2 Evolution of Agile Summary Points

courses.edx.org/courses/course-v1:USMx+ENCE607.1x+3T2019/courseware/eacf1c65220b48bdaa22de2734fc2bf5/9f45cb9162e14eccaffc97baf88c8e4d

TQM: The story of Agile as we know it today begins with Total Quality Management or "TQM," developed by Edward Deming. This was also the origin of the Lean movement that pushed for continuous improvement and appreciation of workers. The core tenants of TQM include:

- Improving Quality Decreases Costs - lowers costly defects, customer support, and recalls
- Continuous Improvement - for the systems and people in the systems
- Pride of Workmanship - the primary driver of knowledge workers and source of quality is joy in good work
- Plan-Do-Check-Act (PDCA) - this cycle allows for testing a complex system that can't be modeled easily

One of the famous beliefs was that the Knowledge Worker is different from the Manual Laborer, because the Knowledge Worker knows more about the work than their boss does.

Proof that TQM Works: Edward Deming turned around Ford Motor Company in 1986 from billions of dollars in losses to its first profits in years using the TQM approach.

TPS: The Toyota Production System (TPS) was developed by Taichii Ohno that was the first true Lean system. Focus was on reducing waste, based on lessons from TQM. The focus was on reducing the wastes in a system:

- Eliminate 7 Wastes - Movement, Inventory, Motion, Waiting, Overproduction, Over-Processing, Defects
- Small Batches - exposes errors and minimizes waste in the system, by using a "Pull System" using Kanban
    - Kanban - means "billboard" and it is a system to tell upstream processes to send work downstream
    - Kanban boards have at least three columns: To-Do, Doing, Done
    - Kanban boards limit work-in-progress (WIP) by limiting the number of items in the "Doing" column and only pulling in more work once the current work in progress is done
- Continuous Improvement with Key Performance Indicators (KPIs)

Proof that TPS Works: Toyota is a top three (3) manufacturer of cars, with a 70% employee satisfaction rating - that's more than double the satisfaction rating of employees in the USA, which stands at 30%.

TOC: The Theory of Constraints (TOC) was developed by Eli Goldratt. It emphasizes that the system is always governed by a bottleneck and there is a competition between local optimization and system (global) optimization. The theory states that Throughput of the system should be the focus of managers, not "Cost Centers" that drive local optimization. His ideas are captured in the famous book The Goal which is read widely and cited as critical to the revolution of management in the 1980s.

- Throughput drives cost and revenue
- Throughput is constrained by one process in any system, the constraint
- To improve the System Throughput one must focus on optimizing around the Constraint
- To do this, use the 5 Focusing Steps for the Process of Ongoing Improvement (POOGI)
    1. Identify the Constraint - figure out which process is limiting
    2. Exploit the Constraint - try to optimize with existing capacity
    3. Subordinate everything to the Constraint - reduce processes to match capacity of the constraint
    4. Elevate the Constraint - add capacity to the constraint process
    5. Prevent inertia from becoming the Constraint - be vigilant and check if there's a new constraint!

Proof that TOC Works: TOC was used by the BP Oil Spill Cleanup initiative to save over $200M and rapidly deploy 10,000 boats after the Gulf Oil Spill to skim and clean oil from the surface.

The Waterfall Mistake

- Waterfall was never intended to be linear in its design
- Royce, who proposed waterfall as a simple starting point for modeling work, stated all projects should iterate
- Typical waterfall design:
    - Requirements - product requirements as output
    - Design - architecture as output
    - Implementation - system is produced
    - Verification - testing is conducted to fix the system where needed
    - Maintenance - support for product in use
- The actual design had at least one iteration going back from verification to implementation to design

Only 10% of software projects were successful in the 1970s, and using waterfall we still see that half of those projects fail today.

By 1980s the Waterfall Method was being used by DoD (and continued until 1996), which resulted in the Ninety-Ninety Rule:

"The first 90 percent of coding accounts for the first 90 percent of development time,
The last 10 percent of coding accounts for the other 90 percent of development time" - Tom Cargill, Bell Labs

Important reference: Waterfall Model Probably the Most Costly Mistake in the World:

http://valueatwork.se/waterfall-model-probably-the-most-costly-mistake-in-the-world/?lang=en

The original paper from Winston Royce on Waterfall Methodology:

Winston W. Royce (1970). "Managing the Development of Large Software Systems" in: Technical Papers of Western Electronic Show and Convention (WesCon) August 25–28, 1970, Los Angeles, USA

Iterative Methods

- Rapid Application Development (RAD) - popular during 1970s and 1980s
    - Upfront requirements
    - Iterate on Design and Development
    - System Cutover
- Dynamic System Design Methodology (DSDM) - popular in 1980s and 1990s
    - Introduced iteration of requirements (foundation)
    - Included going back to design and development as well
    - Still had a "Feasibility Stage" upfront
- Extreme Programming (XP) - popular from 1990s to 2000s
    - Iterative across all levels (Pair programming, unit testing, standups, testing, and planning)
    - Required heavy iterations and redundancy in resources to manage risk
    - Continues to be used today

Key Tenants of Iterative:

- Consolidated Up-Front Planning - RAD and DSDM did not refine, but XP does
- Iterate on Designs - design, build, test, refine
- Timeboxes - to ensure continuous, on-time delivery
- User Stories - to describe requirements (introduced as standards in XP)

- Test-Driven Development (TDD) - enables exploration of designs and refinement before release

2013 Cross-industry Study by Ambysoft shows the following (see the details of the survey here http://www.ambysoft.com/surveys/success2013.html):

- Agile is more successful than Traditional
- Agile is less challenged than Traditional
- Agile fails less than half as often as Traditional

Project Measure	Agile	Traditional
Successful	64%	49%
Challenged	28%	33%
Failing	8%	18%

Note: This table is reproduced from the summary graphics of Ambysoft's survey found here: https://clearcode.cc/blog/agile-vs-waterfall-method/