Lean or Agile:
Lessons Learned from a Tech Startup

HISTORY, CONCEPTS, AND BUSINESS CASE COMPARISONS

BY JOHN JOHNSON, CTO, SOFTEK ENTERPRISES LLC

Agenda

- History of Lean & Agile
  - **The Challenge** - Kelly Johnson, Skunk Works, and the P-80 Fighter Jet (1943)
  - **Lean Movement** - TQM, TPS, and TOC (1950s to Present)
  - **Iterative Development** - RAD, DSDM, and XP (1980s to 2010)
- Lean/Agile Concepts
  - **Basics –** Manifesto and Scrum Mechanics
  - **Science** – Elements and Science
  - **Comparisons**– Agile v. Traditional v. Lean
- Second Nature Software
  - **Background** – Who is Second Nature? What is Rocketfish?
  - **Phase 1** – Going Lean on Product Selection
  - **Phase 2** – Building the MVP with Agile
  - **Phase 3** – Distributing Rocketfish with a Hybrid Approach
- Conclusion

Presenter Profile

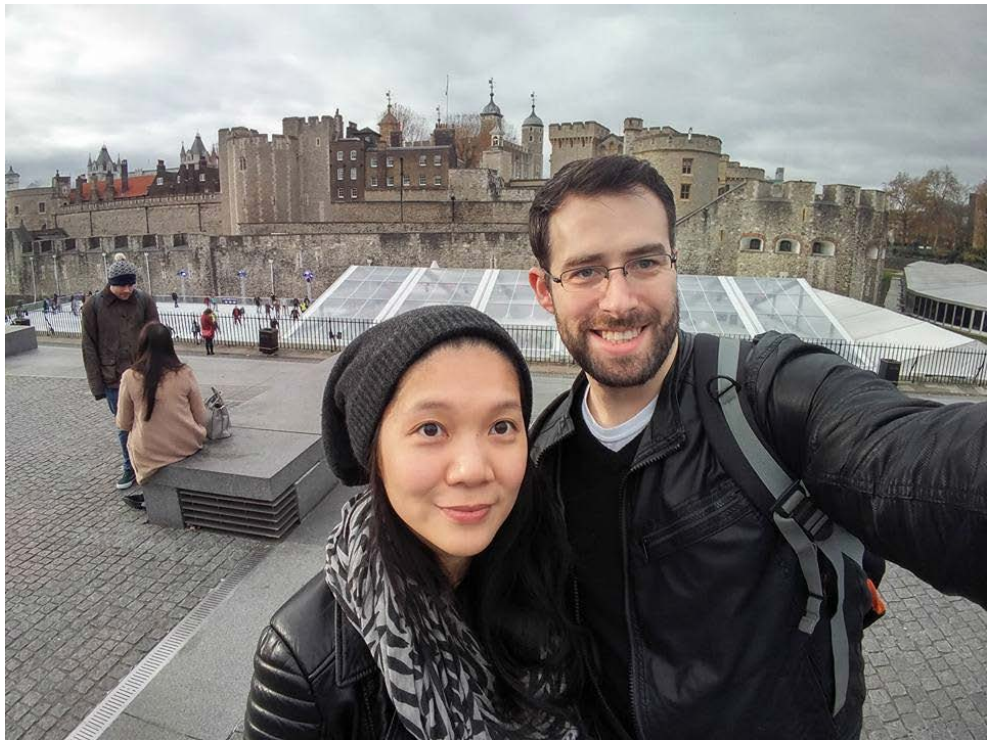
Academic Researcher, UMCP ISR (2008)
- Developed state-based models in LTSA (JAVA)
- Published at INCOSE 2008, no-fault control systems for waterways

Airport Consultant, Ricondo & Associates (2009)
- Planned gate assignments at Dulles Airport (in Excel!)
- Evaluated roads, people movers, airport facilities, etc.

Systems Engineering Consultant, Navy Facilities, BAH (2010)
- Modeling, simulation, and software design for investment tools
- GIS data development, simulation, and modeling for siting tools

Project/Program Manager, CNIC Energy Program, BAH (2012)
- Designed quality control measures to validate investments
- Revamped process & built custom software for investing >$500M/yr

Senior Project Manager, National Archives, IBM (2015)
- Managed development of budget defense, design, and prototypes for new search engine
- Managed teams developing Pilot search engine for world's largest archive by count (award winning)



Co-Founder, Second Nature Software (2016 - Present)
- Identified need for Desktop Data Preparation Tool through Design Thinking Interviews with >30 PIs
- Supported requirements gathering, testing, and design of Desktop Data Preparation Tool, Rocketfish
- Currently supporting organization-wide trial in NIAID and NCI (ask about getting it on your machine if you're in NIAID).

Chief Technology Officer, Softek Enterprises LLC (2017- Present)
- Generated five strategic partnerships in just two weeks with Federal Prime Contractors, Data Science Startups, and SaaS Companies
- Leads marketing strategy overhaul by redefining Softek capabilities in Agile, DevOps, and Cloud technologies
- Manages new product innovation by applying lean/agile principles to design optimal solutions for Softek's clients

Agile Project Delivery Examples

Navy Shore Energy Program, Energy Return on Investment (eROI) Support

Booz Allen Hamilton (BAH)

- Scope:
  - Build decision support systems to identify, evaluate, and select $500M/yr. in shore energy projects
- Total Cost: $5M over 4 years (T&M)
  - 2 Fully Cross-Functional Teams
  - BAH Personnel: 8 (1 PM, 3 Devs, 4 BA/Testers)
  - Navy Personnel: 5 (1 PgM, 3 Officers, 1 Analyst)
- Output:
  - QA/QC avoided $20M/yr. in net-loss projects
  - Improved selection by $30M/yr. annualized returns
  - Modeled investments with 95% accuracy by year 3
  - Project ROI: 50
  - BAH sole sourced the $10M/yr Renewables Program

National Archives Records Administration (NARA), Electronic Records Administration

International Business Machines (IBM)

- Scope:
  - Pilot web-based applications in Amazon Web Services (AWS) for electronic records. Must process, store, and search >100 PB.
- Total Cost: $20M over 1 year (FFP)
  - 6 Fully Cross-Functional Teams
  - IBM Personnel: 35 (15 Devs, 10 BA/Testers, 4 Arch, 4 SAs, 2 PMs)
  - NARA Personnel: 15 (Product Owners, Testers, Architects)
- Output:
  - Delivered on-time and budget
  - Developed a new distributed Agile Method, “Near-Site Agile”
  - First-ever Agile project using all IT Departments (Sys, QC, PM)
  - Project Profit: ~40%
  - “IBM Project of the Year 2016,” out of 4,000 projects globally

History of Lean & Agile

THE CHALLENGE

THE LEAN MOVEMENT

ITERATIVE DEVELOPMENT

The Challenge: Kelly Johnson and the P-80

Clarence Leonard “Kelly” Johnson was a Lockheed Martin Engineer who proved himself in WWII.

In 1943, tasked with extending range of fighter jets.

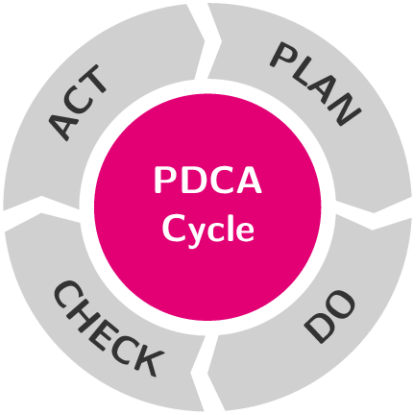
He and his team colocated in a tent because they needed the space… their program was called “Skunk Works” and it did the impossible…

Designed and built the first jet-fighter, “P-80 Shooting Star,” in just 143 days

Kelly Johnson’s Skunkworks Program had ***14 Rules of Management.*** There isn’t enough space here, so I’ve summarized:

- Small, Strong, Self-Directed Cross-functional Teams (1, 2, 3, 7, 13 and 14)
- Owners and Vendors must Collaborate and Trust (7 and 12)
- Mange and Respond to Change (4 and 6)
- Minimize Reports, But Record Important Work (5 and 10)
- Incremental Development with Self-Testing Teams (8, 9, and 11)

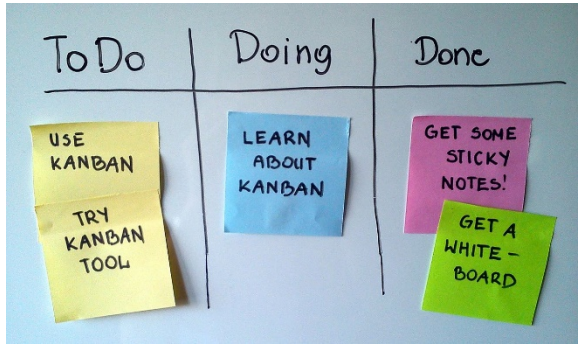
Lean Movement: TQM, TPS, and TOC



Total Quality Management (TQM)– Edward Deming (1950s to 1990s)

- Improving quality decreases costs
- Must continuously improve (systems and people)
- Key is pride of workmanship, cross-functional teams, and trust
- Plan – Do – Check – Act (PDCA)

Proof it works: turned around Ford Motors in 1986 from $B losses to first profits in years



Toyota Production System (TPS) – Taichii Ohno and Lean (1980s to Present)

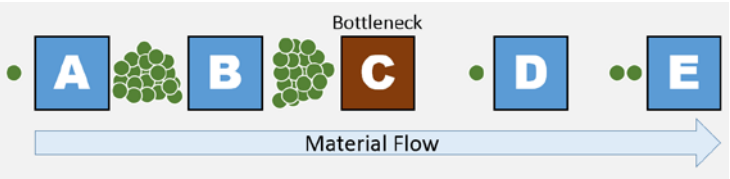
- Eliminate 7 Wastes (Movement, Inventory, Motion, Waiting, Overproduction, Over-processing, Defects)
- Small Batches – addresses most of the waste – *Kanban!*
- Continuous Improvement w/ Fixed Reporting Schedules & Metrics (KPIs)

Proof it works: Toyota's a Top 3 Car Manufacturer with 70% employee satisfaction (vs. 30% avg.)

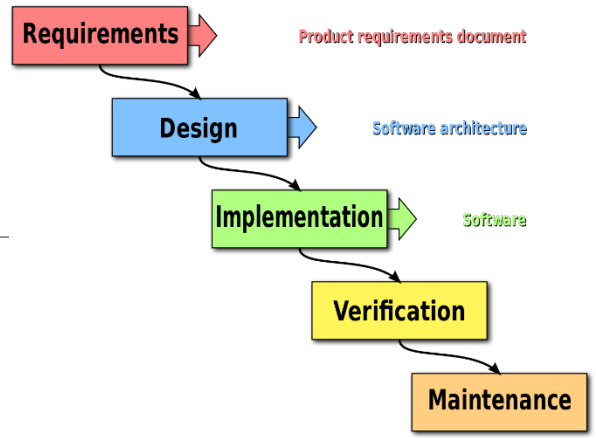
Theory of Constraints (TOC) – Eli Goldratt (1980s to Present)

- Optimize "System Throughput" not "Cost Centers" towards a ***Goal***
- Five Focusing Steps to Exploit System Constraints (Physical, Paradigm, Policy, Market)

Proof it works: BP used TOC to save $200M and rapidly clean 10,000 boats after Gulf Oil Spill



Iterative Development: RAD, DSDM, and XP



By the 1980s "Waterfall" was the predominant methodology, but it was a poor fit for the immaturity of the software development world (although embraced by DoD until 1996)

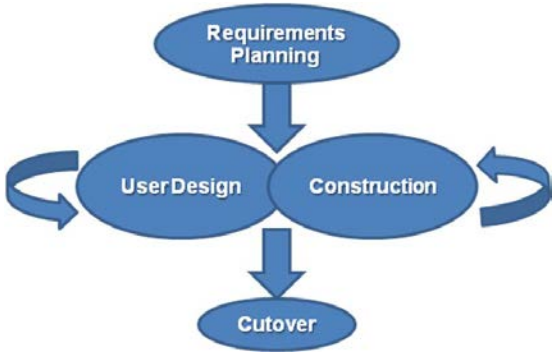
Tom Cargill of Bell Labs said it all with his "Ninety-Ninety" Rule said it all:

*The **first 90 percent** of the code accounts for the **first 90 percent** of the **development time**.*

*The **remaining 10 percent** of the code accounts for **the other 90 percent** of the **development time**.*

In response to failure rates as high as 90%, "iterative development" was born:

- Rapid Application Development (RAD) 1970s and 1980s
- Dynamic System Development Methodology (DSDM) 1980s and 1990s
- Extreme Programming (XP) 1990s and 2000s



Key Tenants of Iterative Development:

- **Consolidated Up-Front Planning** - single "Systems Design" phase with all Stakeholders
- **Iterative Development** – Users Propose and Test Product Throughout Development
- **Timeboxes** - Emphasizes On-Time Delivery
- **User Stories** - Emphasizes Business Needs, Not Tech Specs
- **Test-Driven Development** - Incorporation of "best practices"

***Proof it Works:** Historical surveys from 1970 to 2010 show Iterative projects were 10% more likely to be successful (60% vs. 50%, or 70% vs. 60%). **Value** was also much higher for iterative projects.*

Concepts of Agile

BASICS

SCIENCE

COMPARING AGILE

Lean/Agile Concepts: The Agile Manifesto

Agile was codified in 2001 at the Snowbird Resort by 17 practitioners of Iterative Development. The Agile Manifesto was written by XP, DSDM, and Scrum practitioners stating:

"while there is value in items on the right, we value items on the left more…"

Individuals and Interactions over *processes and tools*

Working Software over *comprehensive documentation*

Customer Collaboration over *contract negotiation*

Responding to Change over *following a plan*

http://agilemanifesto.org/

Lean/Agile Concepts: Seven Wastes ("Muda")

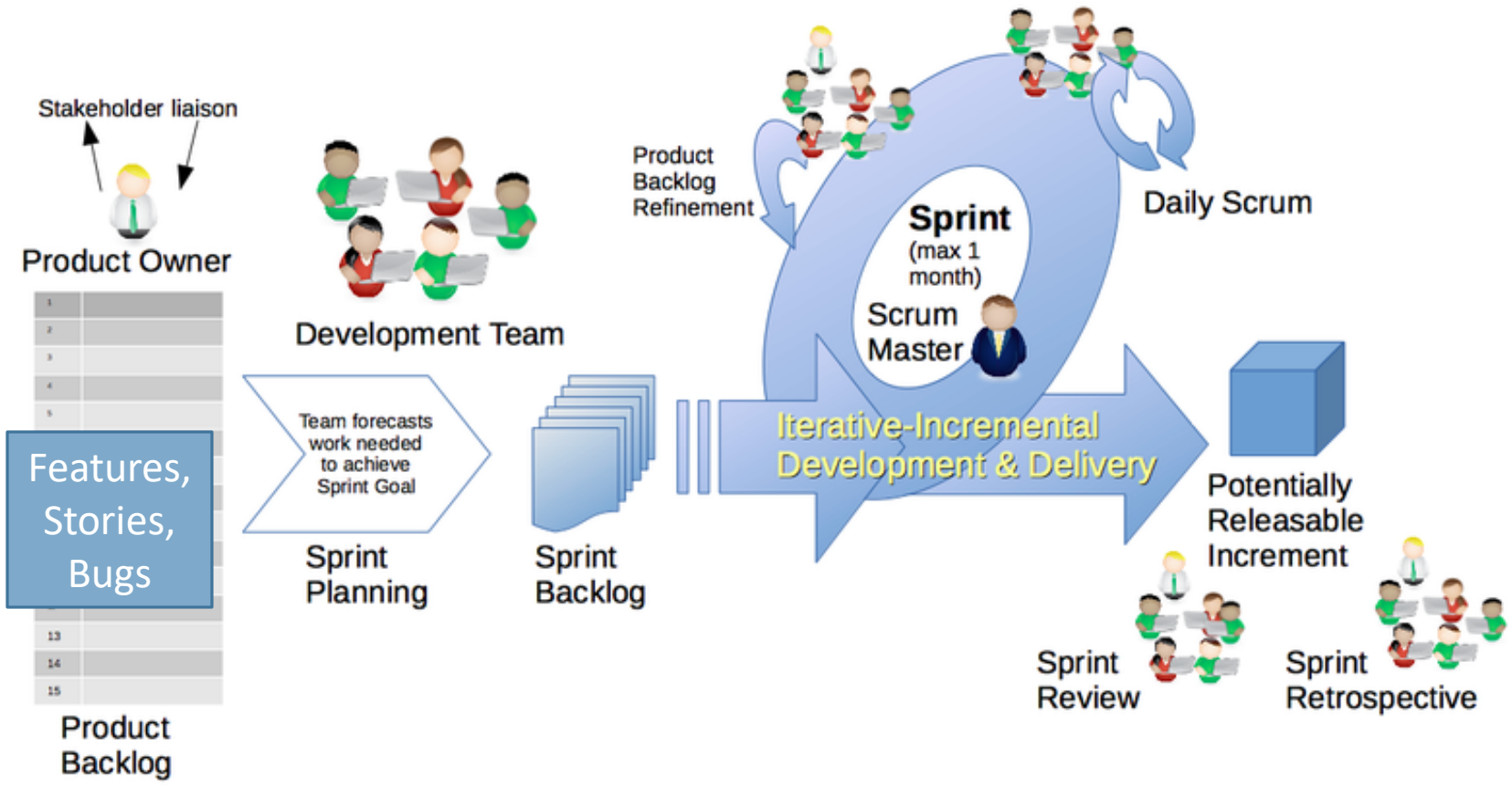
1. **Defects –** minimize potential for defects through simplification
2. **Overproduction** – build only what is needed, then deliver it
3. **Transportation** – reduce movement of product & information
4. **Waiting** – don't let resources be idle, waiting for work
5. **Inventory** – don't let work pile up as it moves across the team
6. **Motion** – keep all work centralized, and optimize workspaces
7. **Processing** – reduce the effort needed to complete the work

Lean/Agile Concepts: Mechanics

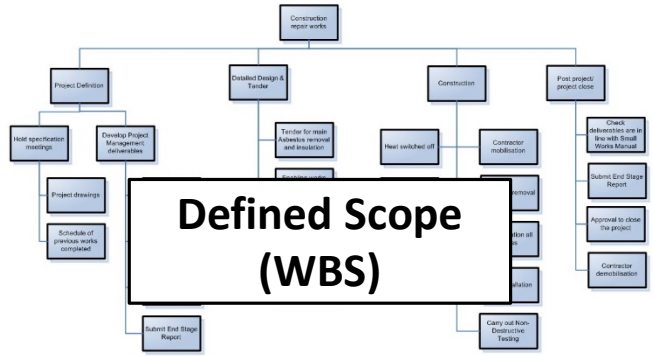
Sprint Planning
- Team & Product Owner select work
- Team commits to complete work inside the Sprint
- All work is stated as a “User Story” with a clear “who, what, why” and acceptance criteria
- Scrum Master facilitates and guides

Sprint Development
- Team meets daily to decompose & assign work
- Team self-organizes based on skills
- No client can interrupt or change their work
- Product Owner liaisons with end users
- Product Owner builds and prioritizes backlog
- Scrum Master facilitates and tracks

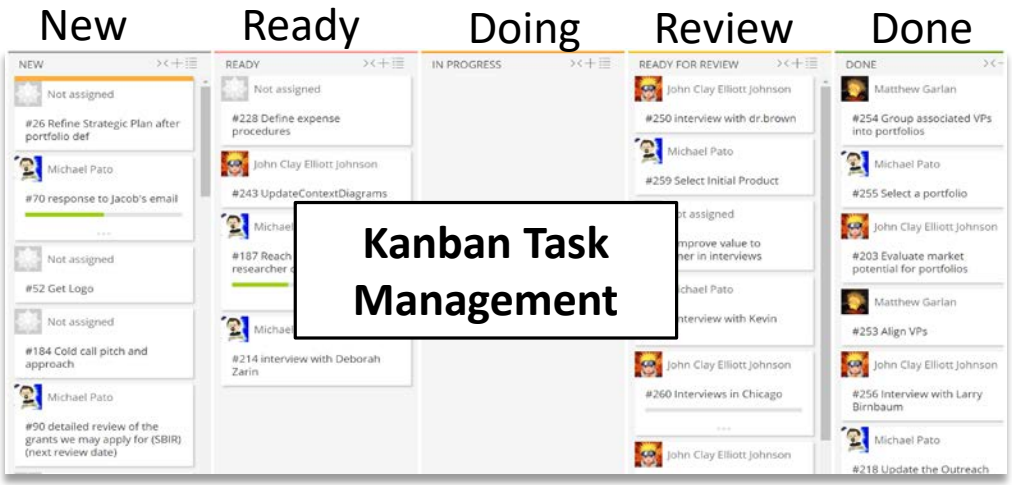
Sprint Review & **Retro**
- Team presents completed work to customer
- Team reviews work performed
- Team performs retrospective to improve itself
- Scrum Master facilitates and guides



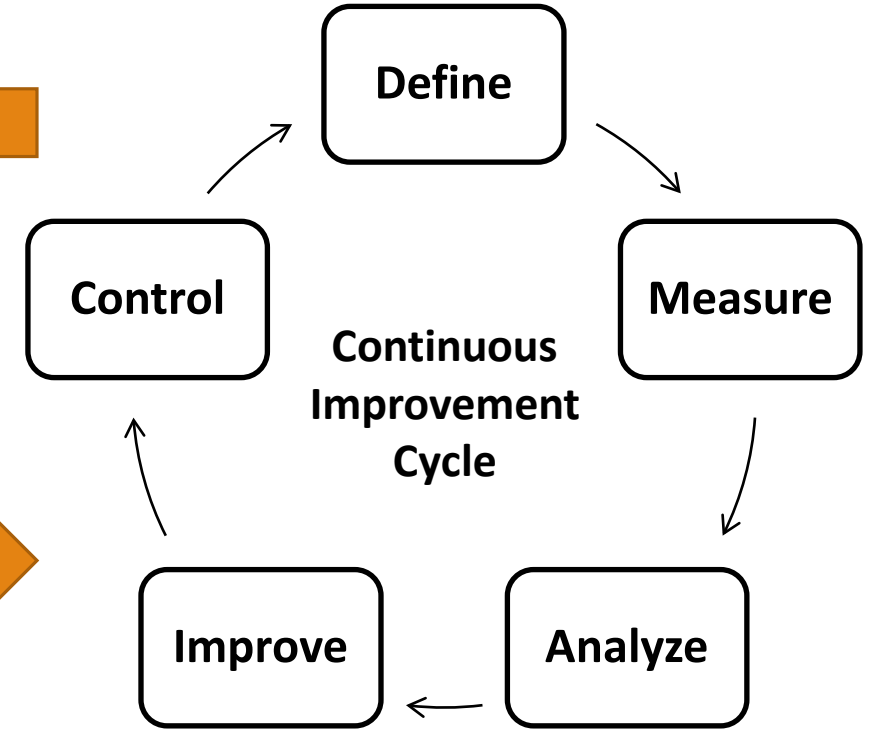
Lean/Agile Concepts: Lean Mechanics



Defined Scope (WBS)



Kanban Task Management



Lean/Agile Concepts: Science

User Stories instead of System Requirement Statements
- Stories require a "who, what, and why" to be complete
- Stories are easy to remember (engages the whole brain)
- Stories increase empathy with users (makes them real)
- Collective agreement and scoping builds pride of ownership

Timeboxes instead of Schedules
- Timeboxing eliminates due dates that cause the "student syndrome"
- Timeboxing allows you to ride the natural variance of tasks
- Timeboxing allows for aggressive "actual" duration estimates of tasks

Iterative and Incremental instead of Planned and Determined
- Iteration speeds up team and project learning by testing designs
- Reduces knowledge gap by doing work in small batches
- Minimizes the cost of changes and delivers working product early

Driving Home? How long will it take…

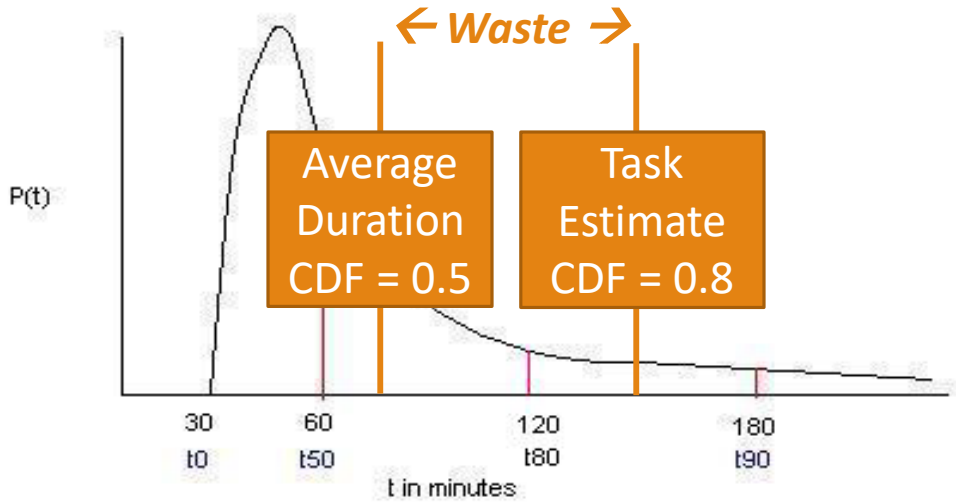
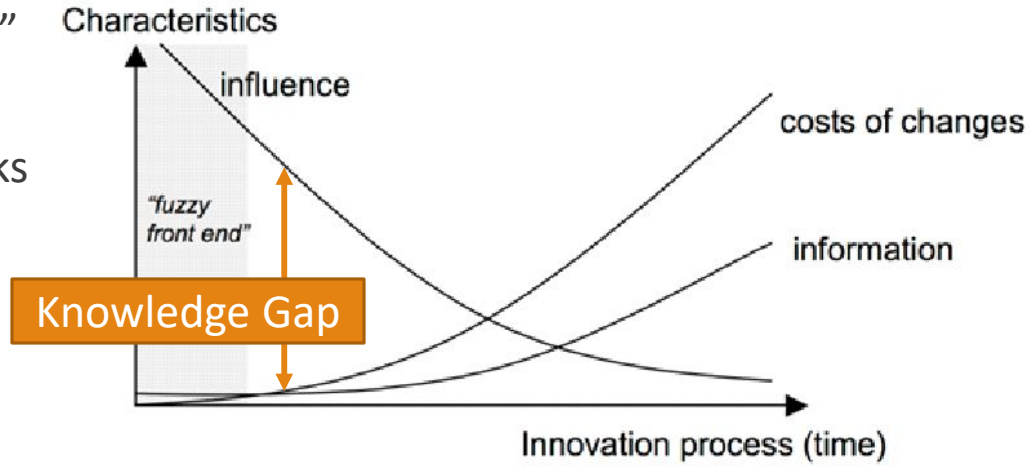
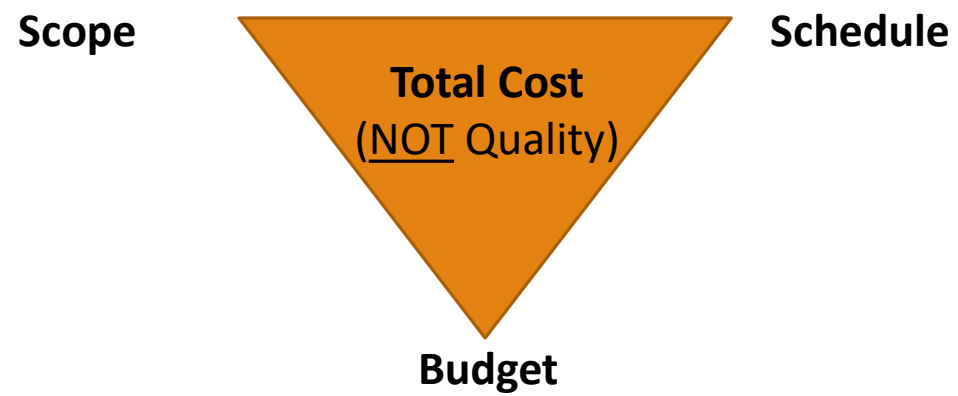


Figure 1 - Probability distribution function for an activity



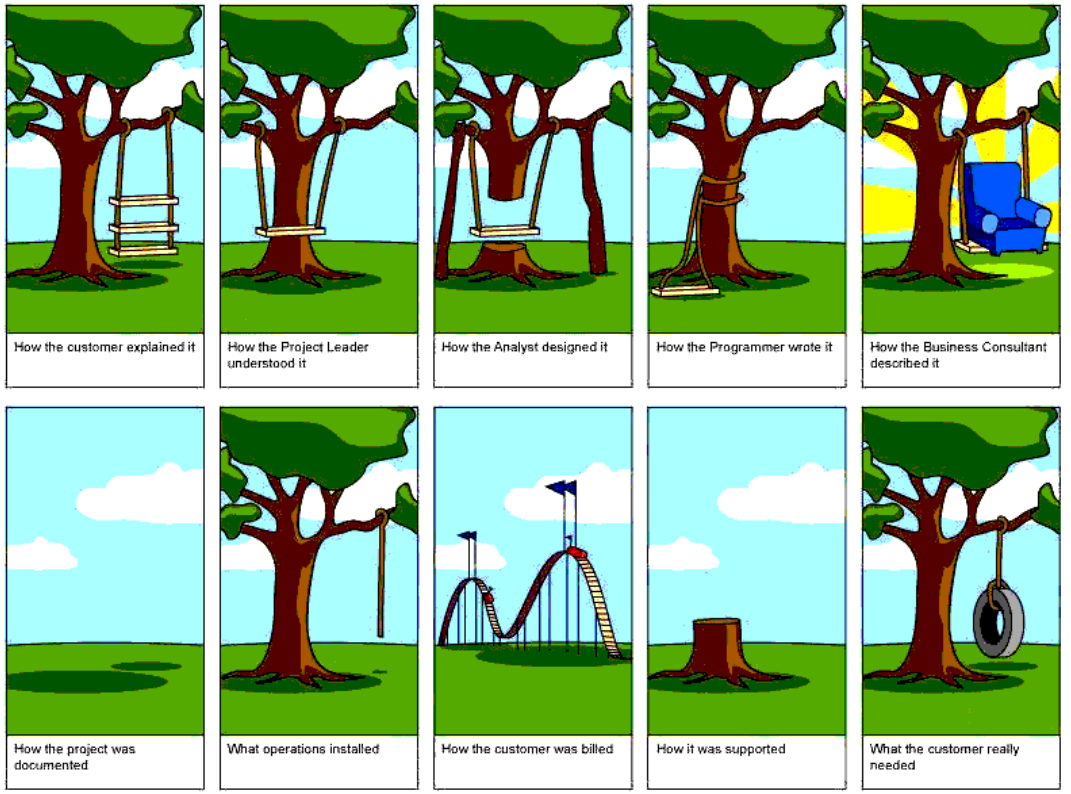
Lean/Agile Concepts: *Managing The Triple Constraint*

Please note (personal opinion):
The Triple Constraint Theory is only a measure of total cost, and not:
- Value
- Quality
- Uncertainty
- Etc.



	Agile	Traditional	Lean
Triple Constraint – what do you adjust?	Scope	Budget	Schedule
Sourcing – how do you pick providers?	Trust	Efficiency	Expertise
Goals – what do you value?	Speed	Predictability	Innovation

Lean/Agile Concepts: False Comparisons



All Methods Have the Following:

- A Charter
- A Plan
- Documentation
- Design
- Testing

Why it's a false comparison

- Projects delivery metrics doe not measure ***Value***
- Projects create value through impacting the organization, at minimum through profits
- Teams are also valuable, and grow faster with continuous improvement
- Knowledge is valuable and can be shared across projects (non-zero sum)
- Time (especially people's time) is the only non-renewable resource
- Agile and Lean deliver faster, tested, working software
- It's easier to collect money when you deliver a working product every 30 days (#realWorldProblems)
- All Projects Experience Changing Requirements – *So Plan for It!*

Lean/Agile Concepts: Points of Failure

- **Lack of overall product design** – most common failure point which is why a good charter, product owner, and "Spike Stories" can help (also a Solution Planning Phase)
- **Adding stories to an iteration in progress** – very easy to do if you under-plan work for a sprint or iteration, or have a lack of collaboration on the team
- **Lack of sponsor support** – teams must be able to get access to end users, make their own decisions, and NOT be interrupted
- **Insufficient training** – this can result in a LOT of stories about learning, or rework, or simply bad planning that builds technical debt. Agile teams must have solid experts to guide them.
- **Product owner role is not properly filled –** very common when organization is not also "Agile." Lack of product owner slows down decision making and, therefore, the team.
- **Teams are not focused** – this happens when you have a wishy-washy product owner, a team that doesn't enjoy their work, or a negative team culture. Quick wins can fix this, though.
- **Excessive preparation/planning** – often happens when there's a lack of trust or experience. Need to push past planning and instead reduce scope to even prototype levels to make progress.
- **Problem-solving in the daily standup** – what a drain! Scrum Masters and PMs beware this time suck and standup killer. Standups are for reporting, team self-organizing, and escalation only.
- **Assigning tasks** – often this means a Scrum Master isn't tracking and asking good questions. Need to ensure when a story is started it is fully tasked before any other work is done.
- **Scrum master as a contributor** – hard to avoid in consulting, but it is necessary if possible. Scrum Masters need to focus on maximizing the team – if they are distracted the whole team suffers.
- **Lack of test automation** – when moving fast on big software it is not possible to test everything manually. Testing automation is essential for maximizing developer hours and team speed.
- **Allowing technical debt to build up** – there are many ways to address technical debt from Separation of concerns to regular "Hardening" sprints to refine and refactor. Often critical long-term.
- **Attempting to take on too much in an iteration** – this happens at first, when team members change, or prioritization has not been effective. Better to miss on too much than not take on enough.
- **Fixed time, resources, scope, and quality** – hybrid methods can cause this to happen to teams. Frankly, every sprint should be clearly defined and the organization must "embrace agile."

Second Nature Software

BACKGROUND

PHASE 1 – GOING LEAN ON PRODUCT SELECTION

PHASE 2 – BUILDING THE MVP WITH AGILE

PHASE 3 – DISTRIBUTING ROCKETFISH WITH A HYBRID APPROACH

Second Nature Software: Background

Second Nature Software's primary product, "Rocketfish," is a data management automation tool that is faster than Excel, while being more traceable than scripting languages like Python or R.

This case study covers the first year of the startup's history and its first three phases of selecting, building, and distributing Rocketfish to the DC-Baltimore medical research community:

- ***Phase 1 - Going Lean on Product Selection***.
- ***Phase 2 - Build the MVP with Disciplined Agile***.
- **Phase 3 - Distributing Rocketfish with a Hybrid Model**.

Mike	***John***	***Matt***
		
Dev Lead	**Agile PM**	**Dev Architect**

Challenge: Build a product company that does social good and supports the team members with:

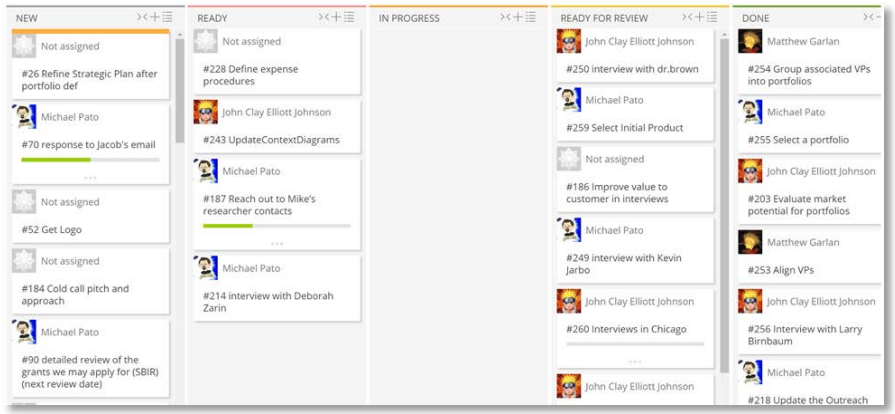
- Three Co-Founders
- No Product
- No Market

Note: At first, the startup was named "One Year LLC" because the team gave itself one year to be successful. However, this name was quickly changed to "Second Nature Software" as the startup did not want any customers thinking the company was unstable.

Second Nature Software:
Phase 1 – Going Lean with Product Selection

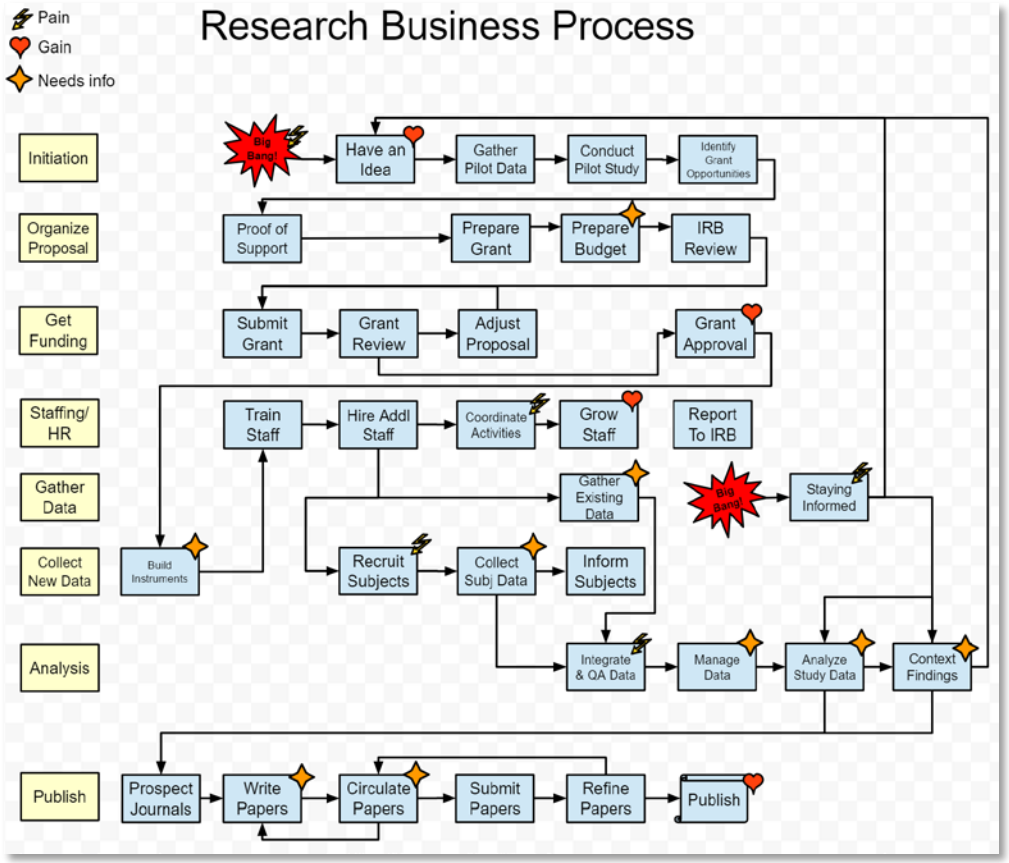
Determining a process to discover product options, drew heavily on Customer Development by Steve Blank and the Opportunity Canvas by Dr. James Green. Using these blueprints the Second Nature team could map out a scope of how to identify a product that best fit the team:

1) Evaluate Team Strengths (Experience, Expertise, and Social Capital)
2) Evaluate Markets that align to Team Strengths
3) Select a Target Market
4) Interview 30 Decision Makers in the Target Market
5) Map Business Process, Archetypes, and Customer Needs (Pain Points and Gain Points)
6) Generate Product Lists
7) Evaluate Products against Team Strengths and Market Needs
8) Select a Product



Markets Identified:
- **Medical Research**
- IT Consulting
- Home Improvement

Products Identified:
- Clinical Trials App
- Literature Review App
- **Data Integration & QA App**



Second Nature Software:
Phase 1 – Lean Analysis & Lessons Learned

Phase 1 Analysis

Efficiently executed, but took too long

- Took 4 months to complete
- Most of the time was spent on interviews
- Interview setup was reduced in half by the end
- Average interviews per week was five
- Startup classes at Stanford were 10x faster (50+/wk)

Many lessons earned that could have been learned

- Most researchers respond to second or third emails
- Referral interviews set up and completed much faster
- Short, casual-language emails with questions more effective
- These lessons are well known by marketing experts

Phase 1 Lessons Learned

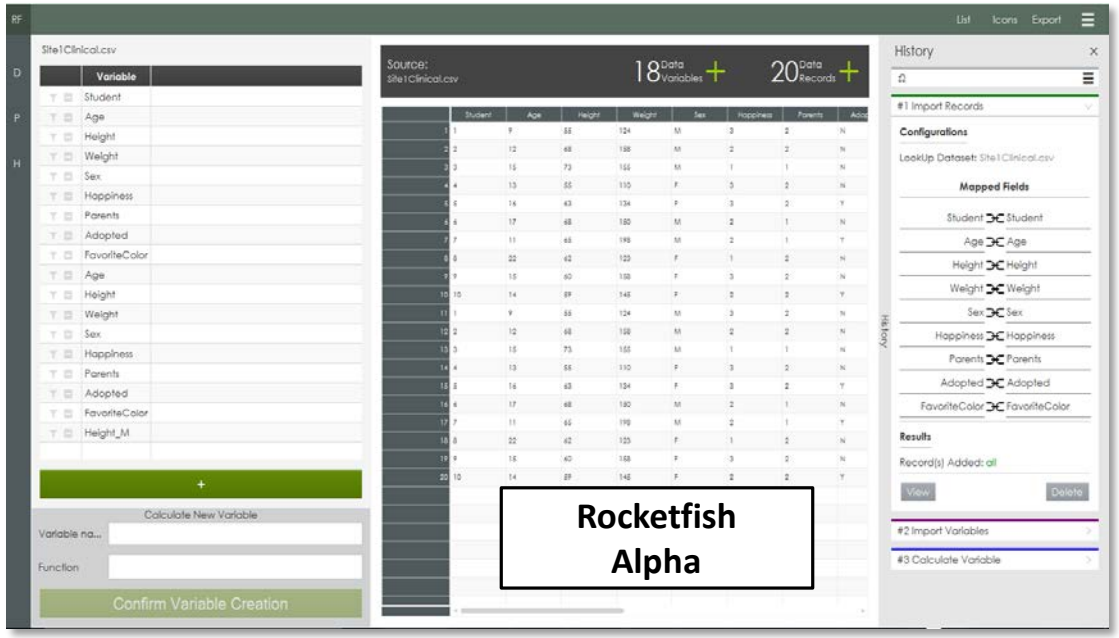
1) **Use Comparable Benchmarks** - leveraging previous, similar projects to understand what would be reasonable to expect in terms of pace and performance.
2) **Seek Expert Advice** - If the team does not have expertise in the work, then the team will waste enormous time in Lean project environments. Lean offers a way to incrementally improve production, but teams without expertise will repeat many beginner mistakes that can be avoided with expert oversight.

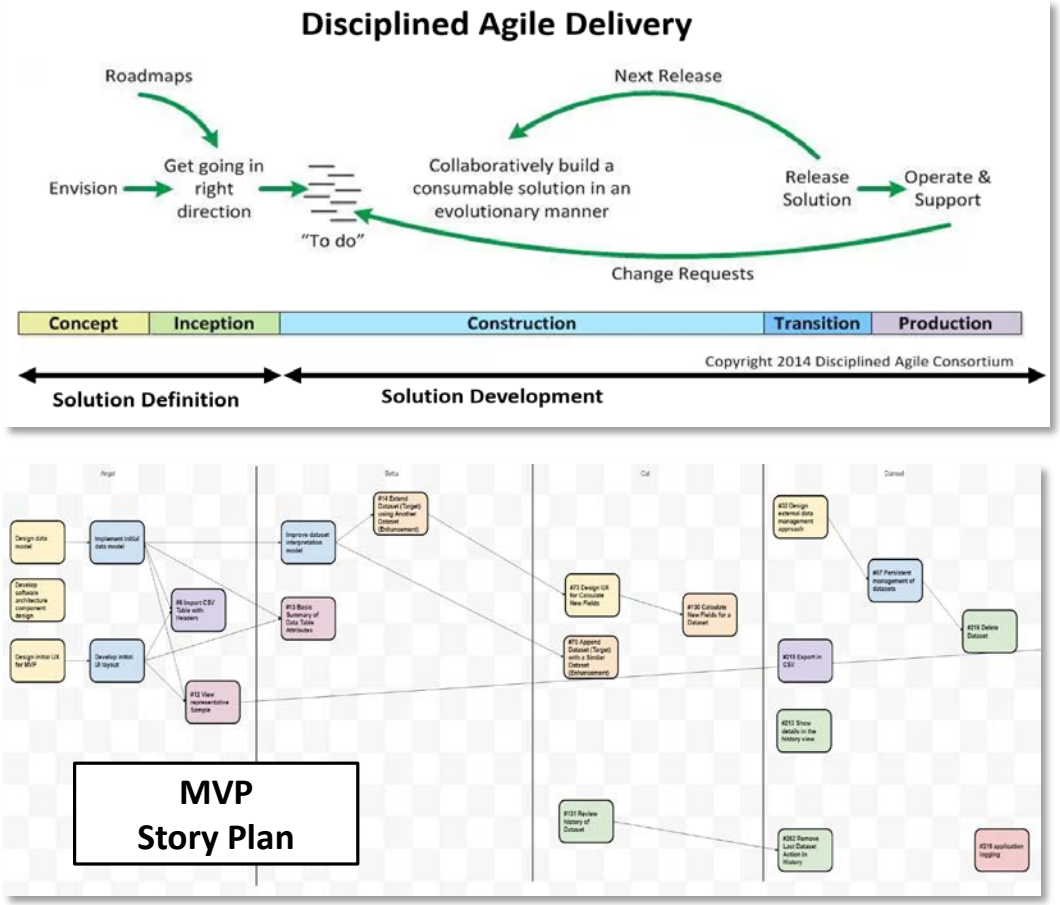
Note: many startups benefit most at early company stages by working in incubators for these reasons.

Second Nature Software:
Phase 2 – Build the MVP with Disciplined Agile

The target early adopters were lab-based researchers who needed to process data collected using assays (test tubes arrays). The features included:

- **Link** - ability to merge datasets with appends (add records) and joins (add variables)
- **Derive** - ability to calculate new variables with existing variables and constants
- **Format** - ability to summarize, filter, and export datasets for analysis





Second Nature Software:
Phase 2 – Agile Analysis & Lessons Learned

Phase 2 Analysis

Delivered a working product quickly, but still rough

- Completed the Alpha version of Rocketfish in two months
- Features were intuitive and easy to use
- 15 evaluators eagerly supported development
- One researcher used Rocketfish to complete days of work in just hours
- The primary sacrifice for efficiency was aesthetics

Feature selection missed critical cleaning and profiling data needs for researchers

- Our data preparation tool needed data prep prior to use
- Data cleaning was required concurrently with key features built, so testing by end users was not possible
- Many researchers needed better visuals for large datasets

Phase 2 Lessons Learned

1) **Emphasize Front-to-Back Design** - start with the user experience in mind and adapt technology to achieve it. There should be no contempt for users, only a focus on delighting them.
2) **Build Features in Workflow Order** - do not build the most important feature first. Instead build the first feature a customer will use first. Know the true bounds of the user's workflow and inject the tool as far upstream as possible to maximize adoption.

Note: this phase was the most enjoyable and landed early wins with trials at NIH. However, feature focus, or "featuritis" caused blind spots to need for data cleaning.

Second Nature Software:
Phase 3 – Delivering Rocketfish with a Hybrid Approach

The team had only enough information for two sprints using Agile to address two major issues:

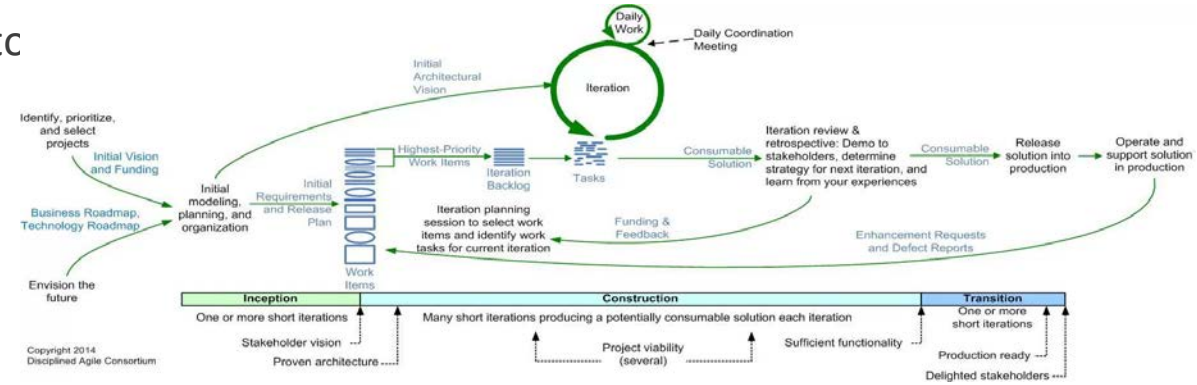
- Users were unable to clean data
- Users needed better visuals and profiling capabilities

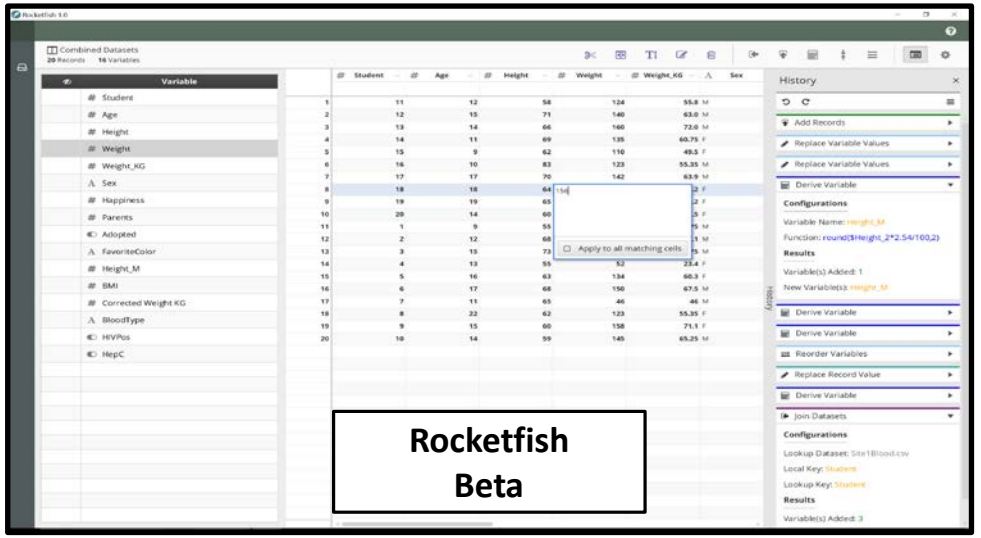
Then the team would need feedback from users for:

- Feature Enhancements
- New Feature Validation
- Defects

The result was a four-month release plan to generate the “Full Beta” with enough features for trials major research organizations:

- National Cancer Institute (NCI)
- National Institute of Allergy and Infectious Diseases (NIAID)
- National Center for Advanced Translational Sciences (NCATS)
- Johns Hopkins University, Bloomberg School of Public Health
- University of Maryland, School of Medicine, Institute of Human Virology
- Rutgers University, Division of Infectious Disease





Second Nature Software:
Phase 3 – Hybrid Analysis & Lessons Learned

Phase 3 Analysis

Hard to balance planned features & customer requests

- Many customers needed specific additional features
- Many small new features were required, such as licensing
- Lack of focus prevented completion of complex features
- New features began to strain the tool with “technical debt”

Application matured but only progressed incrementally

- Rocketfish was continuously improved
- Many loved enhanced features, like advanced calculations
- No revolution to UI or workflow paradigms
- No new differentiating features could be completed such as “visual-based data cleaning”

Phase 3 Lessons Learned

1) **Maintain a Story & Release Plan** - for Lean/Agile needs continuous master planning (Release Plan) and mid-range planning (Story Plan). Otherwise, team members lose direction and motivation.
2) **Plan a Hardening Sprint** - Hardening Sprints are used to clean up technical debt. Knowing the Hardening Sprint is coming adds confidence to build complex stories in packed Lean/Agile sprints.
3) **Build Differentiating Features First** - delaying differentiating capabilities, or “star features,” hurt sales and adoption. Need to ensure these features are still prioritized in packed Lean/Agile sprints

Conclusion

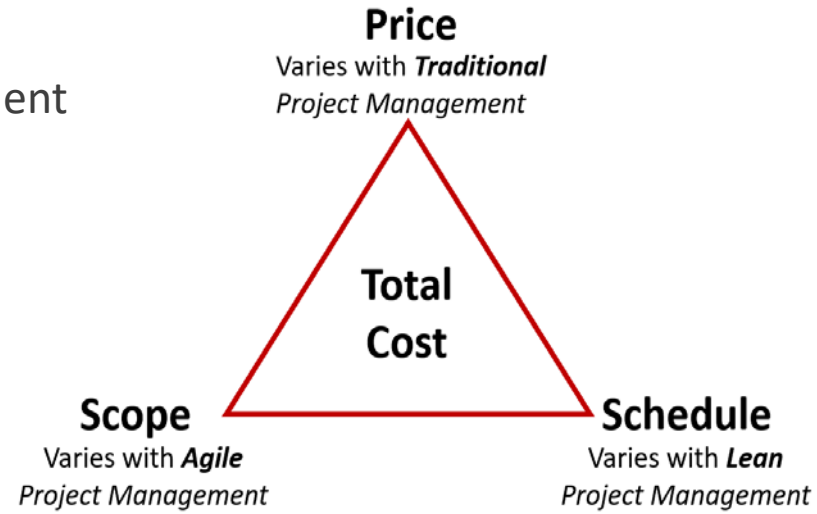
Conclusion:
Lean or Agile: when to use what, how, and why

Agile: *"**What** are the business needs that must be satisfied?"*

- Agile by far is the fastest and most stable for startups
- Works best when asking "what should the team do?"
- Asking "what should we do" works for internal and external work management
- Agile offers the best method for managing *business needs uncertainty*

Lean: *"**How** can the business needs be satisfied efficiently?"*

- Most effective when scope and delivery approach is well-known
- Requires significant team experience in performing similar projects
- Can be extremely efficient for support teams, like DevOps or Architecture
- Lean offers the best method for managing *technical needs uncertainty*



Additional benefit of Agile: answering "***Why"*** as a team

- Agile offers dedicated time for planning as a team at multiple levels (Release, Story, and Sprint Planning)
- Only Agile emphasizes shared team understanding and commitment that creates unity under uncertainty

Thank You and Let's Keep In Touch!

Thank you for participating in this presentation on Lean and Agile.

Please join us tomorrow for the

Panel on Agile Challenges: Procurement & Leadership

Contact Info to Follow-Up on Any Concepts in this Presentation:

John Johnson, PMP

Chief Technology Officer

SOFTEK ENTERPRISES

john.johnson@softekenterprises.com | http://www.softekenterprises.com

412.608.2654 (c) | 443.741.8715 (o) | 267.575.4150 (f)

9250 Bendix Road N., Suite 620

Columbia, MD 21045