Data-Driven Success in Agile Enterprises: EVM and OKRs

John Johnson, PMP CSM SPC
November 2020
A Little About Me…

• I’m a systems engineer turned project manager, who has led at big tech firms, startups, and now I teach!
  - Academic Researcher  UMCP ISR  (2007)
  - Systems Engineer & PM  Booz Allen Hamilton  (2009)
  - Senior Project Manager  IBM  (2015)
  - Co-Founder  Second Nature Software  (2016)
  - Chief Technology Officer  Softek Enterprises LLC  (2017)
  - Strategic Pgm Manager  UMCP PM Center  (2019+)

• In 2018 we launched the 5-course Agile PM Cert on edX, currently has a top 100 MOOC of all time
• In 2020 we launched our edX Product Management Certificate
And our Program at UMD..

Earn Your Competitive Advantage: Two Top-Ranked Programs

Master of Engineering in Project Management (MEng)
- 10 Courses
- 30 Credits
- 400 PDUs

Graduate Certificate in Engineering Project Management (GCEN)
- 4 Courses
- 12 Credits
- 160 PDUs

PM.UMD.EDU
100% Online, 100% On Campus, or Hybrid
Agenda

History of Agile

OKRs & EVM

Summary

Backup
High reuse of components means a “Tightly Coupled” design.

- Small changes have big costs
- Lower costs if no changes
Traditional Management Philosophy

- Traditional or Predictive Management uses operations research and EVM
  - Functional teams self-report progress
  - Work is estimated a % complete
  - Belief that processes are predictable

- This works for standard, well-known processes of engineering

- For novel or innovative projects, in new fields of engineering, this model falls apart
Engineering Issues with Waterfall for Software

• Failure in Physical Systems are gradual
• Failure modes of Software are all or nothing
• Emergent behavior results (mind of its own)
• A truly “wicked” problem
  – The problem is not understood until after the formulation of a solution.
  – Wicked problems have no stopping rule.
  – Solutions to wicked problems are not right or wrong.
  – Every wicked problem is essentially novel and unique.
  – Every solution to a wicked problem is a 'one shot operation.'
  – Wicked problems have no given alternative solutions.
Lean Engineering – Reuse & Refine

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

- RAD 1970s - 1980s
- DSDM 1980s - 1990s
- XP 1990s - 2000s

Lean designs “Evolve” as each feature is developed:

- Uses “just enough” feature by feature
- Reuse as possible to lower costs
- Refine as you learn to address wicked problems
Small Batches Drive Speed
Agile Is Built for Continuous Improvement

Plan → Do → Check → Act

https://commons.wikimedia.org/wiki/File:Scrum_Framework.png
Lean and Agile Systems Broke the Triple Constraint

- **Faster** – Small batches improved speed
- **Better** – Each batch improves the process improving quality
- **Cheaper** – All done with smaller teams

- Iterative development provides “black box” empirical approaches to *wicked problems*

- *But that doesn’t mean we can’t commit to delivery based on existing methods!*
Which Project Lifecycle is Most Suitably Agile?

Spiral Model

1. Determine objectives
2. Identify and resolve risks
3. Development and Test
4. Plan the next iteration

V-Model

Most Agile in Its Design!

Testing

System Validation & Verification

Production Testing
Pilot Testing
User Acceptance Testing
Story Acceptance Tests
Unit Tests
Verify Function
Component Passed
Feature Passed
Capability Utilized
Capability Fully Delivered

Requirements

System Decomposition
System Synthesis
Development

Agile Engineering Emphasizes Discovery

Discover / Develop Customers or Solutions

- Low or no reuse of components means a “Decoupled” design.
  - Small changes have small costs
  - Can release single Features quickly
  - Higher cost if no expected changes

Technology advancements have allowed us to release increments and iterate together

- 2002 – Scrum Alliance Founded
- 2011 – Lean Startup Published
- 2017 – SAFe 4.5 released (w/SAFe 4 Gov)

Low or no reuse of components means a “Decoupled” design.

- Small changes have small costs
- Can release single Features quickly
- Higher cost if no expected changes
Agile Emphasizes Stable, Incremental, and Iterative Designs

### Application Services
- **F1**
- **F2**
- **F3**

**Features**
- **S1**
- **S2**
- **S3**
- **S4**

**High reuse of components means a “Tightly Coupled” design.**
- Small changes have big costs
- Lower costs if no changes

**Lean designs “Evolve” as each feature is developed.**
- Uses “just enough” feature by feature
- Reuses as possible to lower costs
- Often results in complex final product

**Traditional**

**Release in Big Batches!**

**Lean**

**Release in Small Batches!**

**Agile**

**Replace in Small Batches!**

**Low or no reuse of components means a “Decoupled” design.**
- Small changes have small costs
- Can release single Features quickly
- Higher cost if no expected changes
Why do we need to Replace in Small Batches?

*Scale breaks hardware… and Speed breaks software…*

*Speed at Scale Breaks Everything!*  

**Replace in Small Batches!**  
(Microservices)
Can everything be in Small Batches? Well no…

- Small batches work well with construction and implementation of hardware or buildings (e.g. Lean Construction & Last Planner System).

- Small batches don't work well for designing - it should be iterative.

- Therefore, for hardware or big real-asset projects we need prototypes to manage design risk. These can be models (e.g. BIM) or working, non-scalable MVPs.

- Iterative & Predictive approaches are also needed when software and hardware are pushing the performance limits of technology - requiring an integrated design.

- However, most software and hardware innovate on experience or application use, and don’t need integrated designs so they should be Agile.
OKRs Provide a Means for Commitment

- Objectives and key results
  - Objectives are what you’ll do (direction)
  - Key Results are how you’ll do it (measure)

- OKRs come from Intel
  - Developed by Andy Grove
  - Evangelized by John Doer
  - Made popular by Google

- How to develop OKRs:
  - Identify your Flywheel
  - Identify Objectives aligned to your Flywheel
  - Identify Key Results aligned to Project Objectives
  - Use EVM to track Committed OKRs
Spinning the Flywheel – Strategy to Execution

- Flywheel acts as a guide for setting objectives
  - Flywheel are the core differentiators
  - Sets the Innovation Type
  - Each flywheel objective remains long-term

- OKRs
  - Objectives set a measurable outcome that’s timebound
  - Objectives over-time equals speed of turning the flywheel

Mission: Bring Transportation – for everyone, everywhere

- Lower Prices
- More Demand
- Faster Pickups
- Less Driver Downtime
- More Geographical Coverage/ Saturation
- More Drivers

https://medium.com/startup-tools/okrs-5afdc298bc28
Building a Good OKR

- **Objectives**
  - Meaningful
  - Audacious
  - Inspiring
- **Key Results are SMART**
  - Specific
  - Measurable
  - Aggressive
  - Realistic
  - Timebound

### Uber

**Objective 1: Make Uber Driving a Fun, Livable Gig Nationally**
- **KR 1:** Add 20% more drivers per region by end of year (EOY)
- **KR 2:** Increase average driver session to 26 hrs. per week by EOY

**Objective 2: Make Uber the Best Transport Method in Top Cities**
- **KR 1:** Cover 100% of SF and NY metro area by end of quarter
- **KR 2:** Reduce avg. pickup time in SF and NY to < 10 min by end of year

### JJ’s Gym

**Objective: Make JJ’s Gym the best gym in DC**
- **KR 1:** Replace all cardio equipment by the end of the quarter
- **KR 2:** Increase CSAT by 15% in the next quarter based on customer surveys
- **KR 3:** Decrease gym cancellations by 50% per month by end of year
Government Flywheels

- Flywheels also work for Government or “Mission-Driven” Work
- These same strategies can be used to identify the work that needs to be accomplished
- These objectives often are found right on the Government’s website

Mission: The CMS Innovation Center fosters healthcare transformation by finding new ways to pay for and deliver care that can lower costs and improve care.

Lower Prices ➔ Better Care

Improved Experiences/Outcomes ➔ Address Underlying Causes

Lower Costs ➔ Healthier People

Healthier People ➔ Lower Costs

Lower Prices ➔ Improved Experiences/Outcomes

Better Care ➔ Address Underlying Causes

Healthier People ➔ Lower Costs

Lower Prices ➔ Improved Experiences/Outcomes

Better Care ➔ Address Underlying Causes

Healthier People ➔ Lower Costs

Lower Prices ➔ Improved Experiences/Outcomes

Better Care ➔ Address Underlying Causes

Healthier People ➔ Lower Costs
Building and Scaling OKRs

• Types of OKRs
  – Committed – 100% will be met
  – Aspirational – 30%-70% will be met
  – Learning – goal is to learn something

• KR Qualities
  – Input, Output, Outcome
  – Pairing Quality with Quantity
  – Hold, Increment, Leap

• Scaling types
  – Directive – OKRs are cascaded
  – Directional – OKRs are shared publicly
  – ~50% should be bottom-up / directional

<table>
<thead>
<tr>
<th>Company</th>
<th>Directive OKRs</th>
</tr>
</thead>
</table>
| O: Drive up pool sales to $10M | KR1: 12,000 leads  
  KR2: Sign 500 deals  
  KR3: Add upgrade worth $10k |

<table>
<thead>
<tr>
<th>Marketing</th>
<th>Directive OKRs</th>
</tr>
</thead>
</table>
| O: 12k leads  | KR1: CTR 10%  
  KR2: #5 on Google  
  KR3: 20% subscribe |

<table>
<thead>
<tr>
<th>Sales</th>
<th>Directive OKRs</th>
</tr>
</thead>
</table>
| O: Sign 500 deals | KR1: Call 5,000  
  KR2: Quote 1,000  
  KR3: CSAT 90% |

<table>
<thead>
<tr>
<th>R&amp;D</th>
<th>Directive OKRs</th>
</tr>
</thead>
</table>
| O: $10k Upgrade | KR1: Survey 100 users  
  KR2: Sell 100 prototypes  
  KR3: Deploy by Q3 |

Directional OKRs
Turning Key Results into KPIs for Your Product

- **KPI vs OKR**
  - KPI – measures for monitoring performance for teams or products
  - OKR – KPIs that matter to growing or changing the company over time
- **OKRs for Products**
  - Committed – KRs are KPI of team output over time
  - Aspirational – KRs are a KPI goal requiring breakthrough innovation
  - Learning – Testing hypotheses
- **Fits into the Lean Analytics Cycle**
  - KRs draw your line for improvement
  - KRs become Sprint & Release goals

Diagram:
- **Pick a KPI**
  - Draw a line
  - **Find a Potential Improvement**
    - **With data:** Find a Commonality
    - **Without data:** Make a good guess
  - **Did we Move the needle?**
    - **Success**
    - **Pivot or Give Up**
      - **Draw a new line**
        - **Try Again!**
  - **Measure Results**
    - **Design a test (Prototype)**
    - **Make changes in production**
    - Hypothesis
Typical OKR Cycle

- **OKR Planning**
  - Quarterly cycle
  - Q1 Includes Annual
- **4 to 6 weeks out**
  - Brainstorm OKRs
- **Quarterly Planning**
  - Share OKRs w/ Company (2 weeks out)
  - Teams share OKRs (at Quarterly meeting)
  - Employees share OKRs (1-week after)
- **Track Progress**
  - Track KRs for progress
  - Share ~monthly confidence levels
- **Reflect & Score near the end of the Quarter**
Some OKRs Don’t Need EVM

- Use Empowered Agile Teams
- Able to Deploy Regularly
- Testing and Discovering
EVM Fits Big Output OKRs

- Originally, OKRs were used only for output by Andy Grove
  - KRIs are Milestones
  - Milestones establish deliverable output
  - These Deliverables need to be tracked
- Usually these are committed OKRs after good discovery work is complete
“Dotted Line” Teams are teams without delivery responsibility
- Architecture - Architects, designers, marketers*
- Support - Infrastructure, integration, testing

They provide tactical planning, by “pulling forward” requirements, designs, and establishing interfaces

They also provide enabling capabilities like test apparatus or harnesses to integrate work
EVM Fits Teams at Scale for Product Delivery

Customer Discovery
(Concierge MVP / Interviews)

Problem worth solving?
Customer worth having?
Market worth competing in?

Customer Discovery:
- Value Risk
- Usability Risk
- Feasibility Risk
- Viability Risk

Product Discovery
(Hi-Fidelity Prototyping)

Product Delivery
(Featureful Release)

Major Difference in Predictability needed for Delivery and EVM by Tackling Risks Upfront!

Learning OKRs
Outcome OKRs
Output OKRs

Support
Architecture

Build
Prod
Product (MVP)
Meas
Data
Learn
Idea
Learning OKRs
Outcome OKRs
Output OKRs

Customer Discovery:
- Problem worth solving?
- Customer worth having?
- Market worth competing in?

Product Discovery:
- Value Risk
- Usability Risk
- Feasibility Risk
- Viability Risk

Product Delivery:
- Scalability
- Reliability
- Performance
- Maintainability

Value Risk
Usability Risk
Feasibility Risk
Viability Risk

Problem worth solving?
Customer worth having?
Market worth competing in?

Value Risk
Usability Risk
Feasibility Risk
Viability Risk

Scalability
Reliability
Performance
Maintainability

EVM Fits Teams at Scale for Product Delivery
Summary

• History of Agile
  – EVM works well for stable methods, with less uncertainty
  – Lean and Agile work help solve “Wicked Problems”
  – Agile engineering solves to speed @ scale (Speed Wins)

• OKR & EVM
  – Flywheel - the organizational objectives should form a flywheel
  – Measure What Matters - OKRs align objectives with key results
  – Data-Driven Discovery - Learning and Outcome OKRs fit the Lean / Agile
  – Output OKRs & EVM - track large releases of capabilities for featureful delivery
Let’s Talk More at the Symposium!

VIRTUAL
PROJECT MANAGEMENT SYMPOSIUM

APRIL 22–23, 2021

$125 ATTEND LIVE
$150 LIVE PLUS ACCESS TO ALL RECORDED SESSIONS

EARN UP TO 44.25 TOTAL PDUs

4 KEYNOTES
5 TRACKS
55 SESSIONS

PMSYMPONIUM.UMD.EDU
And You Can Learn More Now – Free!

NEW ROLE? NEW PROJECT? WHAT’S NEXT FOR YOU?

- Earn 10 PDUs per course
- Get started now for free
- Verify for only $199 per course

pm.umd.edu/epm
Thank You! Any Questions?

Contact me: johnj85@umd.edu

John C. E. Johnson, PMP CSM SPC.
Strategic Program Manager,
Project Management Center for Excellence

A. James Clark School of Engineering
Civil & Environmental Engineering
1145 Glenn L. Martin Hall
College Park, Maryland 20742
TEL (301) 405-1961; FAX – x 2585;
Email: johnj85@umd.edu
Resilience in PM Methods

- **Predictive**: Fragile
  - Political Reputation
  - Cost-Efficient System
  - Packed Schedules
  - Traditional Software

- **1st Gen Lean**: Robust
  - Worker’s Reputation
  - Redundant System
  - To-Do List
  - Houses and Cars

- **2nd Gen Lean**: Buffered
  - Hero’s Reputation
  - Protected System
  - Prioritized To-Do List
  - Nuclear Reactor

- **Agile**: Anti-Fragile
  - Artist’s Reputation
  - Replicated System
  - Experimentation
  - Roomba
Summary of the Toyota Way
*The Machine that Changed the World*

- Think Long-Term*
- Right Process produces Right Results
  - Transparency and Visual Controls
  - Empower employees w/ Kaizen
  - Create Flow using Kanban
- Add Value by Developing People
  - Grow leaders to live & teach Toyota Way
  - Help improve partners and suppliers
- Continuously Solve Root Problems
The Agile Sprint

https://www.youtube.com/watch?v=O7cA1q0XwhE

Sprint Burndown Chart

https://commons.wikimedia.org/wiki/File:SampleBurndownChart.png

https://commons.wikimedia.org/wiki/File:Scrum_Framework.png
A Costly Mistake

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.

The average Software Project had a 10% success rate in the 1970s

Waterfall model probably the most costly mistake in the world