Table of Contents

Enterprise Lean-Agile Coaching Course

More advanced course on being an enterprise coach.

Context is SAFe as that is where Alex is from so there anti-patterns you hear about in this Training are mainly as a result of consulting in a SAFe environment

Background

From course:

“The Enterprise Lean-Agile Coach Course is a three-day, full-immersion experience for consultants, coaches and leaders whose responsibility is to guide their enterprise through the adoption of Lean and Agile. The course will teach you the practical techniques required to grow the right mindset and culture and to build organizational habits – sustainable practices that match organizational context and continuously evolve to drive maximum enterprise value.

Course topics:

Complex Adaptive Systems

- Organization as a sociotechnical system
- Process as a system of enabling constraints
- Adaptation of practices and dynamic organizational change
- The problem of ‘waterfalling’ the adoption of Agile
- The biggest impediments to adopting Lean and Agile in the enterprise

Mindset and culture

- Diagnosing legacy mindsets in leadership and engineering
- Identifying and removing factors that encourage legacy mindsets
- Aligning and refining enterprise and team mental models
- Factors that form culture
- Growing the right culture
- Anti-patterns in growing mindset and organizational culture

Organizational design

- Organizing around value in complex cases
- Understanding value networks
- Organizational implications of Conway’s Law
- Fixed vs. fluid team structure
- Building a high-performing team-of-teams
- Anti-patterns in organizational design

Building organizational habits

- Making practice a habit
- Impediments to building sustainable practices
- Coaching the right mindset for a practice
- Understanding how practices evolve over time
- Anti-patterns in practice adoption

Building a learning organization

- Implementing validated learning in your organization
- Open and closed-loop systems
- Identifying and addressing blind spots in mental models
- Process evolution in the organization: experiment, scale, repeat
- Anti-patterns in validated learning

Adaptive enterprise economics

- Exploring and defining business value in an organizational context
- Operating with leading and lagging economic indicators
- Optimization within existing funding constraints
- Evolving towards Leaner funding models

Enterprise coaching techniques

- Coaching in ‘Gemba’
- Problem-solving tools
- Building the rollout roadmap
- Coaching vs. training
- Coaching at different levels of the enterprise
- Coaching anti-patterns

Adoption patterns for practices in:

- DevOps
- Architecture
- Quality
- Planning – Synchronization – Demo – Inspect & Adapt
- Exploration and learning
- Metrics and Measures

The course is built around numerous individual and group exercises and simulations designed to effectively explore and internalize specific coaching and change management practices, critical for an enterprise coach.

Prerequisites: This course is for experienced Lean-Agile leaders, coaches and consultants only. This assumes a couple of years of operating in one of the capacities listed above.

The course includes Org Mindset Enterprise Coach (OMEC) certification. Course participants will have the

opportunity to take the exam."

Attendees

Alex Yakyma - http://orgmindset.com

References

Videos

- Eric Ries on "Innovation Accounting" – https://www.youtube.com/watch?v=wdb_6946pGI
- Birds flocking - https://www.youtube.com/watch?v=V-mCuFYfJdI

Dilbert

- Dilbert Key Assumptions for a Business Plan: https://www.youtube.com/watch?v=8s296fATGsU
- Bold New Strategy (reorganize the department): https://www.youtube.com/watch?v=Lv64WFauYLM
- Dogburt Tech Support and bad measures: https://www.youtube.com/watch?v=amnt0Vf1Juc
- Dilbert Accurate numbers (87 studies): https://www.youtube.com/watch?v=A0uBnNHJvfA

Recommended Books

1. N Taleb. Fooled by Randomness (also any other books in "Incerto" series).
2. D Kahnemann. Thinking Fast and Slow.
3. D Reinertsen. The Principles of Product Development Flow. (with caution!!)
4. J Glieck. Chaos.
5. J Humble, et al. Lean Enterprise

Notes

Day 1

Content

1. Intro
2. Complex Adaptive Systems (CAS)
3. Mindset and culture

Important Ideas

- Overall aim when dealing with CAS is to “minimize the constraints associated with collaboration”
- When to use lean vs CAS? If you cannot leverage variability, that's a problem
- Powerful question - litmus test for organization - “How do we treat variability and uncertainty?” If we don't like then we don't understand what is truly going on
- Complex adaptive systems defn “a system, the properties of which cannot be derived from the properties of its components”. This is the exact opposite of reductionist mindset - decompose the whole, learn properties of the parts, derive properties of the whole from the properties of the parts. Reductionist as “how organizations think”. Waterfall is example, but it's everywhere e.g. Planning.
- CAS cannot be directly changed. We can provoke change but only through experimentation
- Characteristics of CAS
    - Emergent Properties: system manifests qualitatively new behaviors that none of its components have (e.g. Ability to evade falcon)
    - Non-determinism: exact outcomes of the system behavior cannot be predicted in principle (i.e. Fundamentally cannot predict)
    - Non-linearity: very small changes may lead to dramatic outcomes (e.g. Sudden transitions, tipping points)
- In CAS context matters
- Don't ignore current reality (e.g. Informal networks)
- Adopt empirical mindset is fundamental - This means worry about improving feedback loops
- Improve feedback efficiency by understanding “feedback marker” e.g. for the system demo might be “do people change their mind after the demo” - source of powerful questions?
- (Action) as coach user feedback markers to understand where the real problems in the system are
- alternative approach to transformation - find cracks / blind spots, then stitch together with feedback loops, then start working to tighten the loops
- (Action) Notion of semi conductor organization anti-pattern. Biased feedback loop always positive. Need to fix with gemba, for example
- (Book) Gerd Gigerenzer - heuristics of decision making
- Implementing agile on traditional mindset means still will end up trying to folllow the master plan. This is because of cognitive bias - risk adverse so don't like uncertainty
- Another set of reductionist thinking that results - thinking point based solutions, thinking outputs not outcomes, chasing predictability over value
- reductionist mental model effects every practice. E.g. Priorization means return / investment. We map to value / size. All good. To to improve value we must experiment with multiple options and validate frequently. This is hard (requires change in mindset). What I can do instead is maximize scope which I understand. Results in deliver more crap faster. Even if you lecture people about this they will still do this. Message is that we manage amount of uncertainty as we move forward. We reduce options as we go.
- Be careful of lean - it's the new cult - anything that is good is lean, but lean is actually (based on tps) manufacturing approach
- (Action) Metrics as ranges always, not a single number
- (Quote) (Sarcasm). “If you are not part of the solution there is plenty of money to be made in

prolonging the problem”
- (Book) The Art of Business Value - Mark Schwertz
- (Action) Get all the videos and distribute / save
- Most important function - spot biases then help people see these. Mindset / mental models
- Law of large numbers does not apply to exponential (long tail) distributions. Only applies to distribution with mean and SD and exponential doesn't have SD
- Any form of planning creates a confirmation bias. Someone has to have an interest in falsify the result (e.g. Dev test relationship - test interested in falsifying)
- Be careful of case studies. Use for exploring but don't apply directly. Case studies work when problem space is normal distribution. And with case studies there is never a control group (therefore confirmation bias)
- Aligning mental models across participants make model more accurate. Two mental models of interest - what to do; how to do it
- (Action) Model alignment technique - identify facts, affinity map / label facts, identify connections, affinity map / label connections. Test model - “where should we focus our efforts” - in other words “use it”. Alternative to true North etc?
- Alignment requires empathy - requires significant overlap in mindset
- Defn -a learning organization is one purposely and collaboratively evolves shared mental models, relying heavily on a system of feedback loops
- You cannot directly change a mindset (as CAS). You can only attenuate factors that encourage wrong mindset, and amplify the ones that discourage wrong mindset.
- (Action) Let understand existing mindsets we need to overcome with SDM

Thinking tools

- Hidden Constraint: helps identify additional areas of flexibility not obvious at first glance. Look for assumptions that are taken for granted
- Hybrid: instead of A vs B, try A and B. E.g. Hybrid fixed and fluid teams
- Endgame: visualize the result and ask questions like why do we need to do this, what do we need to do to get there, how are we going to use it?

Introduction

Mindset is important Sustainability is important

But what should we actually do

Help people to grow the right mindset

Learning is about thinking tools

Goedels theory Any theory, method or model is incomplete Cannot reason about everything in a model There is always a point where you have to step outside the model and learn more

“Keep your eyes open” Constant evolution of ideas

You cannot transfer methods from one organization to another Context matters

Make sure you are solving the right business problem Coffee cup example One is about improving flow (of coffee to the customer) The other is about the experience of the coffee (and flow is not important in this instance)

Context of business matters

Be careful about fixed mindset everywhere E.g. Dichotomy of feature vs component teams It's a continuum We assume the result is fixed when it might not be over time Some situations (e.g. High variability) might not make sense to do fixed teams Purists would argue and there is good data that fixed teams are good in many places, but may not make sense in all cases

Thinking tools Hidden Constraint: helps identify additional areas of flexibility not obvious at first glance. Look for assumptions that are taken for granted E.g.

- Choose between feature and component teams →
- Choose between fixed feature and component teams →
- choose between feature / component orientation and fixed / fluid structure

Thinking tool Hybrid: instead of A vs B, try A and B. E.g. Hybrid fixed and fluid teams Put it on a grid See where intersections are

Thinking tool Endgame: visualize the result and ask questions like why do we need to do this, what do we need to do to get there, how are we going to use it?

Endgame of “improving alignment” could be “we need to make informed local decisions” This is the why Now work on making that happen

Note once you know end state it is hard to not solve the problem With end state in mind you now have inconsistencies as a result of current processes Mind wants to naturally clear up the inconsistency Helps with actually working the issue

We have alignment as a result of ceremonies We still need to work alignment of ad-hoc issues coming up But now we know what problem we are solving

Overall aim when dealing with CAS is to “minimize the constraints associated with collaboration”

Powerful question - litmus test for organization “How do we treat variability and uncertainty?” If we don't like then we don't understand what is truly going on

Training is important but coaching is required to help people through this change in mindset

On going mindset for coaches - explorer mindset - be open minded

Complex Adaptive Systems

Iterative and incremental approaches can be dangerous Reason is that they are easy to implement

Problem is that they can be implemented by using “reductionist” mindset and this approach is not what we need to get to to improve since we are dealing with complex systems

Complete the sentence game 3 iterations, different subjects Then come up advice for a new team that is going to take over your work

Debriefing “why did you come up with different advice?” “Would advice from your team help the other team?” “What are the chances of management coming up with the one perfect work system for both teams?”

Complex adaptive systems defn “a system, the properties of which cannot be derived from the properties of its components” Exact opposite of reductionist mindset - decompose the whole, learn properties of the parts, derive properties of the whole from the properties of the parts

CAS Unique context for each team Once each team sets direction it gets further away from each other over time

Management aim - just enough to get started

This is CAS CAS cannot be directly changed. We can provoke change but only through experimentation

Murmuration of sparrows in Rome 5 million sparrows Avoid falcon - pred/prey Individual bird cannot avoid falcon. Nor one or two. But group has evolved beyond the capability of the falcon Note - bird flying in formation is like this - more efficiency because of non linear properties of the system. This is while it is also more effective for planes to fly in formation

Cas defined by degrees of freedom Number of parameters required to fully describe the system Can model systems - agent based modeling

But cannot model a software organization Too many degrees of freedom But can model parts of a system

CAS change over time E.g. Map number of wolves vs number of deer Observations over time Some systems have an “attractor” around which the phase space of the system will revolve

Sometimes attractor is not single point E.g. Lorenz attractor is like 8 Round and round top part of 8 then something happens and leaps to new state around bottom part of 8 Non linear change in behavior

For nondeterministic systems problems arise when you try to predict things Same with software - it's all about variability Not good or bad, just is

Characteristics of CAS

* Emergent Properties: system manifests qualitatively new behaviors that none of its components have (e.g. Ability to evade falcon) * Non-determinism: exact outcomes of the system behavior cannot be predicted in principle (i.e. Fundamentally cannot predict) * Non-linearity: very small changes may lead to dramatic outcomes (e.g. Sudden transitions, tipping points)

In organization just about all human systems are CAS E.g. Forecasting, adding team member E.g. Flow decreases over time as systems get bigger

Understanding context is therefore important

For organizations people in the system often have hard time to see the context Coaches often have their own baggage of perceived context

Big mistake that people often make is all the informal networks that make things “work around here” And when they put the new system on top, it breaks those informal connection Therefore actually makes things worse And will maybe not recover (or there are new informal networks) And you have created artificial cas

Adopt empirical mindset is fundamental This means to make systems more effective we have to understand the feedback loops of the system

And once you understand the feedback loop understand the “feedback markers” A feedback marker determines feedback efficiency For example, for a system demo often see anti pattern that there is no real feedback In other words people really are just following the plan A feedback marker for the system demo might be “do people change their mind after the demo”

An alternative approach to transformation - find cracks / blind spots, then stitch together with feedback loops, then start working to tighten the loops

We have feedback loops in current organization Semi conductor organization as common anti pattern Good news goes up at speed of light; bad news stays where it is In other words we have an always positive feedback loop Biased feedback is worse than no feedback

Note hierarchy is not good / bad. It just is Problem is the biased feedback loop Idea is to understand / fix the feedback loops E.g. Gemba events (go and see directly)

James Coplien defn of CAS - hierarchy with multiple tops Multiple competing dimensions

Cognitive biases

Book Gerd Gegerenzer

We are risk adverse - cognitive bias

Bias is caused by fact we are uncomfortable if uncertain - mind becomes incoherent Therefore want to become certain again First idea wins whether good or bad So this is hard to educate people on

Result is that we try to implement agile using our traditional mindset We want to execute the master plan We still think point based solutions where there is “a bit of an experiment in the middle” but rest is fixed instead of adopting

Another set of reductionist thinking that results - thinking point based solutions, thinking outputs not outcomes, chasing predictability over value

Be careful of lean - it's the new cult - anything that is good is lean, but lean is actually (based on tps) manufacturing approach

Reductionist mental model effects every practice. E.g. Priorization means return / investment. We map to value / size. All good. To to improve value we must experiment with multiple options and validate frequently. This is hard (requires change in mindset). What I can do instead is maximize scope which I understand. Results in deliver more crap faster. Even if you lecture people about this they will still do this.

Good news is CAS always hides lots of unexplored opportunities to increase value. Bad news is that you don't know about these opportunities early in the plan. So you need to experiment. This is not what people traditionally want.

Problem with Reinertsen view of CoD is that it is assumed that it stays the same in the future

Message is that we manage amount of uncertainty as we move forward. We reduce options as we go.

Idea - develop interface without implementation. This is lean startup thinking

Mindset and Culture

Most important function - spot biases then help people see these

We have mental models - how we perceive the world - what we believe in Impaired by multiple blind spots And we often apply the model to the problem in the first place

“People don't want to think”

Mindset - Some rational (system 2); some emotional (system 1) System 1 will often feed the wrong information to system 2 and then system 2 will rationalize

Idea Law of large numbers does not apply to exponential (long tail) distributions Only applies to Distribution with mean and SD and exponential doesn't have SD

Any form of planning creates a confirmation bias. Someone has to have an interest in falsify the result (e.g. Dev test relationship - test interested in falsifying)

Be careful of case studies. Use for exploring but don't apply directly. Case studies work when problem space is normal distribution. And with case studies there is never a control group (therefore confirmation bias)

This is why Big Bang transformations don't work. It installs foreign mental models which are open to interpretation (to traditional mindset)

Aligning mental models across participants make model more accurate Two mental models of interest - what to do; how to do it Model alignment technique - identify facts, affinity map / label facts, identify connections, affinity map / label connections. Test model - “where should we focus our efforts” - in other words “use it”

Alignment requires empathy - requires significant overlap in mindset

Other approaches to aligning mental models Show by example (in ATDD idea applied to alignment) Cross level meetings (in semi-conductor organization). Needs facilitation - permission to talk.

Seeking alignment, not coercion or conformism People may contradict If do set up experiment to test hypotheses and get the data to validate Goal of alignment is not to force thinking into same template - it's to remove blind spots and uncover new opportunities

Model alignment is not one time deal Learning - but in a way that is controlled

Alignment on model is not enough - Need feedback Else will grow in way that does not reflect reality

Defn A learning organization is one purposely and collaboratively evolves shared mental models, relying heavily on a system of feedback loops

A learning org is empirical and has feedback loops and markers (i.e. These are efficient loops)

Don't rely overly on one form of feedback (e.g. Just demo for product feedback). Have multiple.

You cannot directly change a mindset (as CAS). You can only attenuate factors that encourage wrong mindset, and amplify the ones that discourage wrong mindset.

Let understand existing mindsets we need to overcome with SDM Perhaps create a mindset map - existing to new

Day 2

Content

1. Building Organizational Habits 1. Organizational Structure and Flow

Important Ideas

- Don't turn into a car salesman if you are a change agent - transformation is hard
- Job as coach is to break the cycle - to help manage the fear associated with trying a new practice
- Have to help org leadership understand that new approach may look like being adopted but is not in actuality - need permission to hear bad news
- Building new complex habit is hard. Misunderstand effort to sustain. We Take Big Bang approach instead of more incremental approach. “If you have trouble remembering a process, you are already in trouble”
- Technqiue - relax a constraint on a practice - purposeful

- Sometimes right answer is to not introduce new practice as this time (too much change, too many constraints)
- Minimize practices that have long feedback loops
- We mostly plan before we do research when it should be the other way around - do research first, then do the plan.
- We cannot really reduce value to a single number (e.g. WSJF). In fact, science says this is impossible. Also a problem wrt how to do explorations
- How we view probability - only 3 states none, risky, freak out
- (Action). What is benefit of cadence for storage (does it really make sense)
- People are not machines. To continuously do a practice / process they need to continuously experience the benefits the practice / process offers. I.e. Need continuous feedback
- Need to be empathetic about feedback loop. What does practitioner (Natasha) see?
- If feedback takes a long time to come back, or if action takes a long time, then feedback has no effect. Question - how to make retro more effective. Do action immediately to see if works. Don't wait to next retro to see results
- 5 whys dangerous in CAS as this is not a simple system, but rather complex. Not linear.
- Practice maps show relationships of practices in connected ecosystem. Apply vertical slices approach to implement piece by piece
- (Action) Consider creating practice map next time want to introduce new idea
- We operate in a continuous operate / learn cycle (its own empirical cycle)
- Practice is made up of ways of thinking (practice specific mental model) and ways of doing. So we need to understand the mental model that we need someone to have and then coach to reinforce (sure train as well).
- Different roles people etc will have different mental models they will need e.g. Estimation from viewpoint of developer to viewpoint of manager
- This is way to develop useful catch phrases
- (Action) Make sure I understand mental models we need people to have for practice and catch phrase to use
- (Action) Need to have this coaching discussion with local coaches - get more explicit about what is required from them. Perhaps session by session develop these mental models?
- (Action) Shared cognition. Figure out opportunities to leverage

Thinking tools

- Benefit - constraint: for practice, collaboratively as group wherever possible
- Reinforcing feedback loops: practices need continuous feedback of benefits to become effective
- Practice maps: show relationships of practices in connected ecosystem. Apply vertical slices approach to implement piece by piece
- Embedded mental models: practice is made up of ways of thinking (practice specific mental model) and ways of doing. So we need to understand the mental model that we need someone to have and then coach to reinforce (sure train as well). Model is (class of person) specific (e.g. Dev vs manager). Catch phrases live here.
- Shared Cognition: Leverage others to help learn. We operate in a continuous operate / learn cycle (its own empirical cycle)

Building Organizational Habits

Expectation that we create when we sell transformation - after the sale it will be smooth transformation This expectation stops learning and doesn't allow set up of empirical mindset

Don't turn into a car salesman if you are a change agent

“Black Diamond” initiative story - 20000 people on a quality initiative

Understanding sustainability Practice has its own lifecycle Performance (J curve) Everyone psyched about it, gets hard, drop something critical or revert back to old way of doing things There is no capacity to change the dynamic

Semi conductor org means that management think it's going fine (good news goes up), practitioners say it's all bs and middle managers are protecting in both directions

Internal (coach) SME provides energy to overcome, but this does not sustain

Have to help org leadership understand - need permission to hear bad news

Problem is there is a bad cycle - to try so,thing, need degree of faith to make it work. To make it work need a degree of fair to try it.

Job as coach is to break the cycle - to help manage the fear associated with trying a new practice It's nothing to do with technology or mastery it's all - about emotion

One way is to find someone that has already done it - find the “hot hand”

“No one does test automation in reality”

What happens is we miscalculate our ability to sustain new practice So we say “do it all the time” (e.g. Everything must have automatic test) Instead of making decisions and learning empirically how to apply

Need to make things simple to start with As hard to build new complex habit

“If you have to write the process down to remember it you have already lost” (e.g. Training materials for atdd)

Tools for building organizational habits

- Benefit-constraint
- Reinforcing feedback loops
- Practice maps
- Embedded mental models
- Shared cognition

Benefit-constraint

A different view of a practice - instead of a procedure treat it like a set of enabling constraints (e.g. CI - constraint is “flow of coding frequently interrupted by integrations” or “timebox” is “can't change scope within it, must review work on certain date”

Then can compare benefits vs constraint and decide whether it is worth it

We have thinking bias that all practices are beneficial and remain beneficial in the same way. Not true. May not be and benefit will change over time.

Note there is also meta-constraint on the mindset “How does estimation effect the mindset” For example emergent design meta constraint is false impression of flexibility Cadence is everything has to be done on a cadence, false sense of predictability

Technique Relax a constraint - note this a coaching tool Consider removing or relaxing a constraint in a practice - achieve same with fewer constraints Be careful as become license to do the wrong way Sometimes might be better to just not do the new practice - to much change to be sustainable

Practices constrain each other - need to understand relationship

How does cadence constrain delivery? Hard to detach notion of deploy from “at end of cadence” How does backlog constrain various areas of functionality? Sometime lose big picture to deliver value How does Agile Planning constrain research and exploration? We mostly plan before we do research when it should be the other way around - do research first, then do the plan. Also idea that we cannot really reduce value to a single number (e.g. WSJF). In fact, science says this is impossible. Also a problem wrt how to do explorations

Digging deeper into cadence example Should teams be on same cadence? More about dependencies If two teams cannot do without each other then same cadence. If nothing to do with each other, then perhaps not. Or perhaps there is external dependency on external group and that team should line up with that group Regardless need to explain benefit of cadence

How we view probability Kanneman - only 3 states none, risky, freak out

Reinforcing Feedback Loops

People are not machines. To continuously do a practice / process they need to continuously experience the benefits the practice / process offers

Bad example PMO defines the processes that others use No feedback

Often start practice But then take shortcuts as benefit is not seen (e.g. Walking path)

Learning evolves with the feedback

Need to create / ensure reinforcing loops happen with feedback

Feedback loops tell you something about the feedback loop as well as the system

Need to be empathetic about this “What does Natasha see?”

Two types of practices Ones that have feedback loops automatically built in (e.g. TDD) Ones that need a process (e.g. Estimating)

Pete Seeger in 5th discipline If feedback takes a long time to come back, or if action takes a long time, then feedback has no effect Lagging indicators Question - how to make retro more effective Do action immediately to see if works Don't wait to next retro to see results

Note - 5 whys dangerous in CAS as this is not a simple system, but rather complex Not linear.

Practice Map

Practices for connected ecosystem Create practices map to understand implementation of practice (e.g. This practice dependent on over practice so need some form of other practice in place to get started.)

Apply vertical slices approach to implement

Embedded Mental Models

We don't learn to ride cycles by training. We learn by doing. More complex practices require experiential learning.

So a practice is made up of ways of thinking (practice specific mental model) and ways of doing. In particular, the values, principles etc are not “somewhere in a cloud”. So we need to understand the mental model that we need someone to have and then coach to reinforce (sure train as well).

We operate in a continuous operate / learn cycle (its own empirical cycle)

Turn into a series of catch phrases Fewer is better than more

E.g. Estimation is a reduction but not elimination of uncertainty

Takeaway - training without subsequent coaching is useless

Need operate learn cycle Need hypothesis Need way to validate that hypothesis - what is experiment

Shared Cognition

When we work with people we have a different (better) experience E.g. Reading news vs watching news E.g. Customers who bought also liked E.g. Stock market (everyone is selling Facebook)

Figure out ways to leverage Instead of person, team, or groups etc

Interactions amplify learning

E.g. Community of practice about “shared cognition”

Ideas

Delegation of responsibility - during pi planning evens, SoS, etc Root cause analysis bias - bring in red team, split one problem to two groups

Idea - once you see something work, hold onto it, and use it to build something else Happens with natural systems all the time

Day 3

Content

1. Organization Structure and Flow
2. The System Under Development
3. Adaptive Economics

Important Ideas

- Once you see something work, hold onto it, and use it to build something else
- Problems with value streams as org principle for teams -Too big, Too small, Too convoluted, Variable business demand: value stream looks this way one day, different the next
- Defn Value stream can go in any sustainable direction in which we increment value
- Work team organization through dependency structure made up of IVA
    - I - team a depends on team b
    - V - platform and two consumers
    - A - two teams supplying one before then can complete work
- Hub and spoke model flow visualization
    - Kanban board - Product/Feature, Team A, Team B, etc each has todo WIP and done.
    - CFD on feature and each team (related) - arrival feature starts when any team starts on feature, departure feature finishes when all teams have finished
    - Slice features across teams to validate incremental progress
- (Action) Hub / spoke model flow visualization for PI planning?
- Understand the effect medium you are working with has on economics - benefit of pair programming is that the medium is not code (tangible, slower, after the fact of being made

concrete) but rather communication (intangible, faster, upfront). Applies to other things as well - any time there is a tight relationship
- For org (or any design) keep system open for intangible and closed for tangible connections

For intangible want to spread information - actively communicate e.g. Using cop, Yammer, wiki, virtual conference

- Hierarchy of (business) demand being fulfilled by (team / capacity) supply through a portfolio (of products).
- Cannot reduce priority to a single number (e.g. CoD). Idea to use Story Mapping (Patton) / Impact mapping (Adzic) approach to reason about multi-product / smaller number of teams to get some of value
- Need more lean startup organization to leverage variability. Organization we want is about setting up an environment where an organization establishes hypotheses and validates ideas, and then takes those validated ideas to create business value
- Sunk cost is a strong cognitive bias
- Move org from “scope driven” to “lean startup”
- (Action) Need to think about Kotter in this context. Two operating systems vs change mindset
- (Action) Need to figure out mental model, feedback loops, benefit / constraint for lean startup style org
- If we don't understand the System Under development we cannot understand the complex system. Start by understanding dependencies within / without
- (Action) What do I need to understand about storage System Under development?
- Need to help people combat linear thinking. Use Monte Carlo simulation to model and show results.
- There really are 10x people. Need to understand where they are in the dependency of work.
- Determine what, if anything to do about Lehmans laws
- Idea - develop a “change heat map” to reason about the changes in a system. Go through backlog and say “what modules does this effect” until we have a clear understanding of what impact we see with changes
- Architecture role - faster, more stable delivery of value. Design flow optimized for responsiveness, variability, change at different role
- Traditionally architecture is way too big and takes way too long. Shrink cycle through slicing
- (Blog) “Essential skills of an agile developer” - Alan Shalloway
- (Blog) “Emergent Design” - Scott
- Watch being “fooled by randomness” (load teams up with features simulation)
- Watch “thinking I know the answer”
- Sometimes the right answer with a system is to do nothing
- (Write Blog) about this
- Need to understand side effects of measures as no measure is neutral
- Develop measures based in “what business hypotheses are we trying to validate” and develop proxy to learn this (as most measures lag) -
- (Video) “Innovation accounting” with Eric Ries

Thinking Tools

- Static - Dynamic mismatch: consider two interacting parts over time - is one side changing, the other not (e.g. Demand and team structure)

- Tangible - Intangible connections: For org (or any design) keep system open for intangible and closed for tangible connections. For intangible want to spread information - actively communicate e.g. Using cop, Yammer, wiki, virtual conference
- Demand - Portfolio - Supply model for organizations
- Monte Carlo simulation to help understand system better

Organization Structure and Flow

Idea - once you see something work, hold onto it, and use it to build something else Happens with natural systems all the time Pyramidal systems and extra pyramidal systems Hold hand out, bend one finger down. How does it work. Instruction to bend whole hand of fingers, second instructions to have 3 fingers disregard the instruction Evolution of extra system - only mammals have it.

Perhaps don't have to get too bold

Organizing to Deliver Value (Definition)

Want to identify flows of value too set up teams / org structure

Problems (or combinations of)

- Too big: too many people to reason about the org
- Too small: too many things to small group of people
- Too convoluted: not all value flows left to right, in isolated value streams
- Variable business demand: value stream looks this way one day, different the next

We can abstract flow of work Idea - design / NFr - code - integrate - test - deploy Code means “change functionality” no matter what level you talk about it. Integration means bring different implementations together (logical branches) to ensure it works

This is true whether operating at team or feature level.

Value stream can go in any sustainable direction in which we increment value

Number of fallacies about value stream:

- Product means value stream: may not as may not be incrementing value (e.g. Anyone touching PowerPoint?)
- Functionality that cuts across multiple products, its temporary and therefore just backlog item (e.g. Microsoft move to cloud had no additional features but effected every product for 2 years)
- Business value == customer value and we should organize around customer value (e.g. Customer is input, but business needs to make money)
- If we organize architecture around value, business demand will stand still (not it will change)

Thinking tool Static - Dynamic mismatch: consider two interacting parts over time - is one side changing, the other not (e.g. Demand and team structure)

Means we always have to asses the time impact on things as well

What if we have two sustainable backlogs with common PO and with associated teams but one team supplies both orgs This is bottleneck, conflicting priorities etc Theory of constraints says have to focus on bottleneck. Right answer might be to stop sending backlog items to “common team” to allow them to get control of WIP. Rest of system is constrained by this team so need to work the constraint

Organizational Patterns

Approach

Organize teams around value Based on patterns Observe and experiment, design, emergence (sometimes get good unintended results)

Start with dependency map (of teams) Have teams Track the development of a feature through the team Do this with a number of features See where thickest lines are - dependencies Now zoom in on these Is there the same pattern? Perhaps need to reorganize to create a new team base on this repeating pattern

From Goldratt - Critical Chain In general patterns of dependency fall into three basic structures (IVA):

- I - team a depends on team b
- V - platform and two consumers
- A - two teams supplying one before then can complete work

For V Can we restructure If person on team is doing work for another, perhaps they should be on that team May still leave some people

Why doesn't value stream mapping work for a team? Problem is that whole there is a flow to work (requirements, design, code, test) these don't happen consistently all the time.

Idea - hub and spoke model Teams contribute to value delivered (hub is value, spoke is teams)

This is idea of feature team thinking

Mental model of Kanban board:

- Left to right
- Manage WIP

Doesn't have to be left to right: Hub and spoke flow visualization - Product/Feature, Team A, Team B, etc each has todo WIP and done. Can quickly see feature and team WIP, also progress on features, etc CFD on feature and each team (related) - arrival feature starts when any team starts on feature, departure feature finishes when all teams have finished Slice features across teams to validate incremental progress Makes SoS meaningful - aimed at shared delivery of features

Choice of Medium

Why do we think pair- / mob-programming work

Think about the problem of integration with 2 people working on code. If we integrate, there will be problems and these will take time to resolve This is because we have already made thoughts “concrete” through code - it's tangible

What if we paired? It would take us a lot less time to integrate as we would talk about things before they become concrete The medium is not code (tangible) but rather communication (intangible)

Message: Understand the effect medium you are working with has on economics - benefit of pair programming is that the medium is not code (tangible, slower, after the fact of being made concrete) but rather communication (intangible, faster, upfront). Applies to other things as well - any time there is a tight relationship

It is economically good decision to pair Applies to many things If it takes time to resolve a dependency that you could have picked up in a PI planning event through discussion then wouldn't that be better

Open / Closed Principle

Applied to organizations

Idea is to encapsulate dependencies within “requirements area” - LESS term

Different types of connections

- Tangible - Hard: Team A needs something from Team B
- Tangible - Software: Teams don't need functionality but integrate into same code base
- Intangible - inconsistencies don't lead to delivery issues but increase entropy and misalignment (e.g. Two teams implement “fuzzy search”; symptom god class)

For org (or any design) keep system open for intangible and closed for tangible connections For intangible want to spread information - actively communicate e.g. Using cop, Yammer, wiki, virtual conference

Demand - Portfolio - Supply model

Thinking tool

Hierarchy of (business) demand being fulfilled by (team / capacity) supply through a portfolio (of products)

Simple case: Demand → Product → Multiple teams implement This would mean a single backlog for teams

But not always this way: e.g. Demand → 3 products → implemented by two teams Could mean no single

backlog as cannot easily figure out priority order

For these, need to be able to reason about what is released Use Story Mapping (Patton) / Impact mapping (Adzic) approach Not just a capacity budget as you need to make sure you get something “whole” across all the products

Defining Value

Note. Cannot reduce priority to a single number (e.g. CoD). Need more lexigraphical ordering process based on different levers (e.g. Strategic themes)

This is general scientific result:

All planning is speculation Goal is not interactive and incremental delivery of scope

Need to move away from “Agile” as a means to develop faster (waterfall wrapped in iterative / incremental ritual)

Need more lean startup approach to leverage variability Organization we want is about setting up an environment where an organization establishes hypotheses and validates ideas, and then takes those validated ideas to create business value

Sunk cost is a strong cognitive bias almost as strong as risk aversion

How do we break the loop Perhaps start with doing something simple - pilot on slicing feature and changing direction to produce value. Then generalize

Need to figure out mental model, feedback loops, benefit / constraint for lean startup style org

Other practices that drive “scope-driven” mindset

- Fixed forecast / fix scope projects
- promising before we have a capacity discussion
- just saying “yes”
- regulatory compliance interpretation
- capitalization interpretation

The System Under Development

If we don't understand the System Under development we cannot understand the complex system (as it effects the total system)

If software then could look at dependency map of code

The link between variability and complexity

Help understand test economics If fix bug, there is chance that some other related module is effected and so on This is a complex system in its own right Simulating this out you get long tail distribution Problems is that with our linear thinking we think “I fix bug and sometime that means I break something (e.g. p=.2 or 20% of the time) else” But real expected value of this distribution is 3.8x times what this kind of think would tell you when you model this 20% probability with Monte Carlo simulation. This is because the 20% might not just effect 1 additional node, but two or three other modules (based on understanding of dependency graph)

What does this tell you about your estimates for bug fixing?

What effects “p” in your system:

- Unit test
- Cyclomatic complexity
- Branching
- Knowledge sharing (intangible)
- Length of time with branch checked out (more chance of regression)

Performance of people

Increasing performance from “The Best and the Rest: Revisiting the Norm of Normality” - O'Boyle and Aguinis Chart basically shows there really are a few 10X performers in a number of industries, just not done for software Yakyma doing it for software based on “number of commits” for developer.

Two problems with this

- What will happen if low performer at the node of the dependency network
- Expectation is that high performers try to control what they work on and so won't be on that critical dependent node

Lehman Laws

- The System Under development must be continually adapted or it becomes progressively less satisfactory
- As a system evolves, its complexity increases unless work is done to maintain or reduce it
- As a system LLC associated with it, developers, sales personnel and users, for example, must maintain mastery of content and behavior to achieve satisfactory evolution. Excessive growth diminishes the mastery. Hence the average incremental growth remains invariant as the system evolves.
- The quality of the system will appear to be declining unless it is rigorously maintained and adapted to operational environment changes
- The system evolution process constitute multi-level, multi-loop, multi-agent feedback systems and

must be treated as such to achieve significant improvement over any reasonable base.

Note - below is another source these are different to the ones quoted by Alex. There is also a more formal version.

- The law of continuing change. – Any software system used in the real-world must change or become less and less useful in that environment.
- The law of increasing complexity. – As time flows forwards, entropy increases. That is, as a program evolves, its structure will become more complex. Just as in physics, this effect can, through great cost, be negated in the short term.
- The law of large program evolution. – Program evolution is a self-regulating process and measurements of system attributes such as size, time between releases, number of reported errors, etc., reveal statistically significant trends and invariances.
- The law of organizational stability. – Over the lifetime of a program, the rate of development of that program is approximately constant and independent of the resources devoted to system development.
- The law of conservation of familiarity. – Over the lifetime of a system, the incremental system change in each release is approximately constant.

Idea - develop a “change heat map” to reason about the changes in a system Easy approach is to go through backlog and say “what modules does this effect” until we have a clear understanding of what impact we see with changes

Traditionally architecture is way too big and takes way too long. Shrink cycle through slicing “Essential skills of an agile developer” - Alan Shalloway “Emergent Design” - Scott

Adaptive Economics

Economic prioritization and capacity

Defn - economic prioritization is any form of sequencing of backlog items that produces improved cumulative economic outcomes By defn requires it to be time and value aware

Exercise Teams A, B, C, D, E List of features in priority order Effort distribution on each feature shows effort of each team to complete feature Think effort unit as team week of work

Load features to fill grid for first 10 weeks of work

Seems to show Team E is overloaded Theory of Constraints says we should do something about it But this is not lean manufacturing - variability is part of what we do Today Team E is overlaoded but may not be tomorrow. Best way to deal with this is to bring work we can to teams with T-shaped skills not reorg teams

Note to me Watch being “fooled by randomness” (load teams up with features simulation) Watch “thinking I know the answer” Sometimes the right answer with a system is to do nothing

Measures

No such things as neutral measure

Therefore for each metric need to determine

* Motivation it is expected to induce * Side effects * How to avoid those side effects

Develop measures based in “what business hypotheses are we trying to validate”

Course, SAFe, Learning, Coach, Consultant, Enterprise