Table of Contents

Premise

Are you trying to scale your agile project to a program, a collection of projects with one strategic objective? If you do what you've done with one small project, you'll get bloat. Instead of bloat or large frameworks, you can use agile and lean approaches to manage your program with small-world networks. Small world networks help each team to remain autonomous, and still collaborate and explore across the program.

The common risks for software programs are how to manage the interdependencies, how to nurture the architecture, how to see the status, and how to release an entire product. When we ask feature teams to collaborate and take responsibility across the organization, the teams can manage many of the interdependency and architecture challenges. With program management, we can see the status and release the entire product.

Learning Outcomes: * How to scale out, not up, to create effective programs * How the teams can self-organize to small-world networks * How deliverable based planning and short deliverables can help a program with agile and non-agile teams * How the program manager is a servant leader and what the program manager does

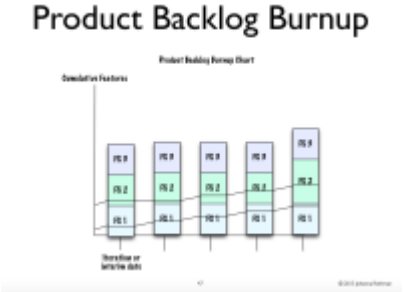
Attachments: Agile_Program_Management.ScalingCollaboration.key.pdf

Summary

- Content rating (0-no new ideas, 5 - a new ideas/approach, 9-new ideas): 7
- Style rating (0-average presentstion, 5 - my level, 9-I learned something about presenting): 5

Action / Learning

- Book - predicting unpredictable
- Book - agile and lean program management
- Understand how see, understand and encourage small world networks at work
- Review epic progress chart (one chart with everything)



Presentation

agile_program_management.scalingcollaboration.key.pdf

Jr@jrothman.com @johannarothman Www.jrothman.com

Notes

1988 program - lisp board

What worked -

Clear vision Clear deliverables Feature teams that delivered at least every day Unit tests and systems tests to support collaboration Respect / trust across the organization - eg everyone worked on code base

Program manager - Organize coord several projects Don't tell - make it possible to coord themselves Program of concurrent projects

Programs riskier than projects Big effort, more risk More pieces, more risk Projects don't scale linearly

Types of program Integrated system program - eg smartphone, embedded systems Interrelated programs - operating system and related systems, base products / layered products

Programs How often can you release - SaaS - continuous, every day Boxed software Product with firmware Software with hardware or mechanical components

SaaS means architectural decisions and releasing costs less - make decision as late as possible (this is last responsible moment). Perhaps no sprint zero for architecture Hardware means these decisions cost more, last responsible moment is early

Servant leaders Ask for results you want Ask for estimates, don't mandate Ask for bad news as soon as possible (want to figure out when I need to be involved) Facilitate problem solving where the problems are (not through the hierarchy)

Cynafin framework Complex - unknown unknowns (don't know what's going to get you) - what experiments we need to run Make things resilient Small chunks of planning

How do I make my problem resilient (not how do I prevent problems) Problem is that governance - many come from waterfall days

Scaling agile requires autonomy (on feature teams), collaboration (across teams), exploration

Agile roadmap 6 quarter release plan Wish list - no bearing on reality 1 quarter perspective But hard for teams to plan this out and then stick with them

Planning for 1 or 2 iterations Then use rolling plan

Do for teams, and do for programs

Do you need a 6 month program instead of a 6 quarter roadmap

How often can you release Internally - get it to “every month”

How small can we make our stories

Rank things by value - sometime learning about the risk is valuable

How can I make my program able to change, often Feedback often Change often

Program teams

Solve problems across the organization that the teams cannot solve themselves

Core team - shepherds business value of the product Includes - legal, deployment, marketing, finance etc Program manager talks to everyone else Might also have chief (program) product owner etc

Like Kanban for the core team Reason - people don't understand things finish things if don't understand agile So put Kanban board See action lists

Software program team manages obstacles for feature teams Software program manager Not one from each team Look at feature sets and see which ones are related This is where communities of practice need

Like Kanban board here as well

Agile program management Team manage Commitment How build features Evolve arch

Program management Remove obstacles Collects / explain program status

Small team able to produce on regular basis

Scale out, not up.

What the most effective way to move information in your organization

You have a network in your org - rumor mill You don't have to mandate communication with rumor mill Powerful

It's a small world network

Ask for rumors at every meeting Can use this network

Small world networks work but don't need to have everyone know everyone else

Programs take advantage of network so they don't need hierarchies Because people will talk to each other

Ask people for results you want Then see what the problems are and facilitate resolution

Continuous integration - so I can demonstrate this at the end of month What other options do you have Then see the results.

Always optimize for the total product the total program Not for the team

Allowed to have the time to help others succeed Use small world networks

I would like to see this in two weeks time

Organize the teams - but doesn't really mean do the same thing What matter is that teams release on a regular basis Small batch size small. - not more than a day long

Evolve the architecture

See the flow of work over the organization Manage wip

Use small world network

Feature teams take responsibility Prefer feature team

Experiment if you have component team - try feature team

Use this to drive communities of practice

Collaboration across the organization Not up and down to solve problems Don't call it “Chiefs” - don't have rank Hierarchy slows everything down

Transparent progress Transparent / pervasive communications

Teams collaborate with tests and integration Have systems tests → give to developers day one Unit test as well Plus exploratory

Estimating a program

You might need a high level estimate of everything The farther out it is, the worse the estimate date

What can you do? Work toward a target date or cost How much do you want to invest before we stop

Rank you value Question - should we be doing this at all Business value points Cost of delay - what if we don't do it now Waste

More often you rank / update the roadmap, the more teams will hunk about delivering value

Delivery solves many problems The more teams deliver, the more often you can update the roadmap, the more you get feedback, the more you re-rank the next backlog

Culture of delivery running tested features solve many interdependency issues Swarm across team

See program progress Program level measures Working product is best measure Don't use velocity - use budget?

Epic 1, epic 2, etc with burnup per iteration Stack of epics showing total, with burnup for each epic

Think small to go large Small batch size Technical practices essential - ci, unit tests,system tests support change and help deliver finished features

Release often - see progress, invite collaboration Frequent releases encourage exploration It's agile / lean all the way across

Reporting, Scaling, Conference